

15-1293

**United States Court of Appeals
for the Federal Circuit**

RICKY D. HANGARTNER.,
Plaintiff – Appellant,

v.

INTEL CORPORATION, INC.,
Defendant – Appellee.

*Appeal from the United States District Court from the District of Oregon
in Case No. 13-cv-00141, Judge Michael W. Mosman*

OPENING BRIEF OF PLAINTIFF – APPELLANT

PHILIP P. MANN
TIMOTHY J. BILICK
MANN LAW GROUP
1218 Third Avenue, Suite 1809
Seattle, Washington 98101
(206) 436-0900

JOHN WHITAKER
WHITAKER LAW GROUP
1218 Third Avenue, Suite 1809
Seattle, Washington 98101
(206) 436-8500

*Attorneys for Plaintiff – Appellant
Ricky D. Hangartner*

April 30, 2015

CERTIFICATE OF INTEREST

Counsel for Appellant Ricky Hangartner, certifies the following:

1. The full name of every party or amicus represented by me is:

Ricky D. Hangartner.

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) represented by me is:

Ricky D. Hangartner.

3. The parent companies, subsidiaries (except wholly-owned subsidiaries), and affiliates that have issued shares to the public, of the party or amicus represented by me are:

None.

4. The name of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or agency or are expected to appear in this court are:

MANN LAW GROUP
Philip P. Mann
Timothy J. Billick

WHITAKER LAW GROUP
John Whitaker

MANSFIELD LAW
Johnathan E Mansfield

Dated: April 30, 2015.

Respectfully submitted,

/s/ Philip P. Mann
Philip P. Mann

TABLE OF CONTENTS

I.	STATEMENT OF RELATED CASES.....	1
II.	APPELLATE JURISDICTIONAL STATEMENT.....	2
III.	STATEMENT OF THE ISSUES.....	3
IV.	STATEMENT OF THE CASE.....	4
V.	STATEMENT OF THE RELEVANT FACTS.....	5
A.	Background of Dr. Hangartner.....	5
B.	Summary of the ‘422 Patent.....	6
C.	Intel’s Accused Microprocessor Product.....	7
VI.	SUMMARY OF THE ARGUMENT.....	9
VII.	ARGUMENT.....	10
A.	Standard of Review.....	10
B.	Claim 1 Should Properly Be Construed To Recite “ <i>One Or More</i> ” Logic Elements.....	10
1.	“Logic Element” Is Introduced As Singular Or Plural.....	10
2.	Latter Uses Of “Logic Element” Necessarily Refer To Initial Use Of “Logic Element”.....	11
3.	Latter Uses Of Same Term Maintain Same Plural Nature Of Initial Claim Term.....	12
C.	The Claim Construction of “Logic Elements”.....	13
1.	The Competing Claim Constructions.....	14
2.	The District Court’s Flawed Linguistic Claim Construction Analysis.....	15
3.	The District Court’s Flawed Prosecution History Analysis.....	16

4. The District Court’s Flawed Reliance On The Problem Solved By The Inventions Of Claims 2-10.....	18
VIII. CONCLUSION.....	21

TABLE OF AUTHORITIES

TABLE OF CASES

Teva Pharma. USA, Inc. v. Sandoz, Inc., 574 U.S. ____ (2015).....	9
Markman v. Westview Instruments, Inc., 517 U.S. 370 (1996).....	9
Cybor Corp. v. FAS Techs., Inc., 138 F.3d 1448, 1454 (Fed. Cir. 1998).....	9
Energizer Holdings Inc. v. Int’l Trade Comm’n, 435 F.3d 1366 (Fed. Cir. 2006).....	10
Papyrus Technology Corp. v. New York Stock Exchange, Inc., 581 F.Supp.2d 502, 533 (S.D.N.Y. 2008).....	10
Baldwin Graphic Systems, Inc. v. Siebert, 512 F.3d 1338, 1342-43 (Fed. Cir. 2008).....	12
Liebel-Flarsheim Co. v. Medrad, Inc., 358 F.3d 898, 906 (Fed. Cir. 2004)....	17
Environmental Designs, Ltd. v. Union Oil Co., 713 F.2d 693, 699 (Fed. Cir. 1983).....	18
Neomagic Corp. v. Trident Microsystems, Inc., 287 F.3d 1062, 1075 (Fed. Cir. 2002).....	19

TABLE OF STATUTES

28 U.S.C. § 1338(a).....	2
28 U.S.C. § 1295(a)(1).....	2

I. STATEMENT OF RELATED CASES

Counsel is unaware of any other appeal in or from the same civil action or proceeding as this matter that was previously before this or any other appellate court.

There are no other cases known to counsel pending in this or any other court that will directly affect or be directly affected by this Court's decision in the pending appeal.

II. APPELLATE JURISDICTIONAL STATEMENT

(a) Jurisdiction in the District Court was based upon 28 U.S.C. § 1338(a) because this is a civil action arising under an Act of Congress relating to patents.

(b) This Court's jurisdiction is based on 28 U.S.C. § 1295(a)(1), this being an appeal from a final decision of a District Court in a civil action arising under an Act of Congress relating to patents.

(c) This appeal is timely under Fed. R. App. P. 4. Final judgment was entered by the District Court on January 5, 2015. A Notice of Appeal to this Court was timely filed on January 23, 2015.

III. STATEMENT OF THE ISSUES

1. Did the District Court err by construing Claim 1 of the '422 Patent to require “multiple logic elements.”

IV. STATEMENT OF THE CASE

Appellant Ricky Hangartner brought this action for patent infringement originally in the Western District of Wisconsin on September 19, 2013. On motion by Intel Corporation, Inc., the Western District of Wisconsin transferred the matter to the District of Oregon on January 27, 2014. The parties conducted claim construction on a number of disputed claim terms, which the parties believed would likely be case-dispositive.

The parties exchanged opening and responsive claim construction briefs, and a claim construction hearing was held on November 17, 2014 before Judge Mosman of the District of Oregon. After that hearing, the District Court issued an Order construing the claims of the '422 Patent. Based on the District Court's claim construction order, the parties jointly agreed that Dr. Hangartner could not make an infringement case based on the District Court's claim construction. Accordingly, the parties jointly moved for a stipulated Final Judgment of non-infringement so that the District Court's claim construction order could be reviewed by this Court. The District Court granted that joint motion, and this appeal ensued.

V. STATEMENT OF THE RELEVANT FACTS

This is an action for infringement of U.S. Patent No. 6,464,422 (“the ‘422 Patent”), entitled “Probabilistic Computing Methods And Apparatus,” issued on October 8, 2002. The action was brought by Dr. Ricky Hangartner against Intel Corporation, Inc.

A. Background of Dr. Hangartner

Dr. Hangartner is a research scientist with a doctorate in computer science. (A081). Using his knowledge and experience, Dr. Hangartner conceived and developed a unique and highly effective way of attempting to solve extremely difficult “combinatorial computing problems” using a “probabilistic computing system.” *Id.* One aspect of solving a combinatorial computing problem involves generating random boolean values as a “guess” at a solution to the combinatorial computing problem. *Id.* As part of his work, Dr. Hangartner developed an extremely efficient random number generator used as a subsystem of the probabilistic computing system. *Id.*

In early 1997, Dr. Hangartner met with Intel's Strategic CAD group, including Dr. Carl-Johan Seger, to interview for possible employment with Intel. *Id.* During those meetings, Dr. Hangartner and

Intel's employees discussed in great detail the technology Dr. Hangartner had developed and was then patenting. *Id.*

B. Summary of the '422 Patent

The '422 Patent is generally directed to a probabilistic computing system wherein random numbers are generated as part of proposed solutions to complex computing problems. (A082, A030-051). A key feature of the preferred embodiment is a "nondeterministic subsystem" that efficiently generates random numbers. (A044, Col. 2, lines 20-36). Other subsystems of the probabilistic computing system are a "testing" circuit and a "feedback" circuit. *Id.*

The '422 Patent includes ten claims, three of which are independent and seven of which are dependent. (A049-051). The several claims define three distinct families of claims. Two of the claim families, which include claims two through ten, are directed to the entire "probabilistic computing system" expressed as circuits for solving highly complex problems through the use of random values as proposed solutions. The second family, which is only claim one, is directed only to the nondeterministic subsystem expressed as a circuit for generating random boolean values. *Id.*

Independent Claim 1, the only asserted claim, is reproduced here for the convenience of the Court:

1. A nondeterministic logic circuit for generating random boolean values of one or more variables as a proposed solution to a computing problem expressed in conjunctive normal form as one more clauses in said one or more variables, the logic circuit comprising:
 - (a) **one nondeterministic logic element** for generating a respective random boolean value for each one of the said one or more variables; and
 - each nondeterministic logic element comprising:
 - a cross-coupled pair of transistor inverter circuits; means for controlling power to the cross-coupled pair of transistor inverter circuits; and
 - means for equalizing charge on the gates of the transistor inverter circuits while power is removed from the cross-coupled pair, thereby driving the cross-coupled pair to an unstable equilibrium, whereby intrinsic circuit noise will cause the cross-coupled pair to randomly assume one of two stable states when power is restored to the cross-coupled pair, the stable state assumed by the cross-coupled pair providing a probabilistically selected random boolean value and further comprising common synchronization means
 - (b) coupled to all of **the nondeterministic logic elements** for synchronizing operation of the nondeterministic logic elements.

(A49-50)(emphasis added).

C. Intel's Accused Microprocessor Product

Defendant Intel is a leading manufacturer of semiconductor chips used in computers, mobile devices, and other systems distributed and used throughout the world. (A084). Starting in about April of 2012,

Intel introduced to the marketplace several of its latest series of microprocessors code named “Ivy Bridge,” “Haswell,” and “Silvermont” (collectively the “Intel processors”). *Id.*

Incorporated into the Intel processors are circuitry and an instruction set that, among other things, enable users of the Intel processors to generate random numbers. *Id.* The ability to generate random numbers is important to users of the processors and is a valuable feature of the Intel processors. *Id.* The Intel processors generate random numbers through the use of circuitry and techniques developed by Dr. Hangartner and disclosed and claimed in his ‘422 Patent.

VI. SUMMARY OF THE ARGUMENT

The District Court erred in its construction of Claim 1 of the ‘422 Patent by failing to appreciate that the term “logic elements” in the latter portion of the claim was only plural because it refers back to the introduction of the term where it undeniably refers to “one or more logic elements.”

The language of Claim 1 undeniably introduces the term “logic element” as envisioning “one or more logic elements.” When the term “logic elements” is used later in the claim, it undeniably refers back to the original “one or more logic elements.” The District Court improperly focused on the plural nature of the latter term “logic elements” and ignored the undeniable reference to “one or more logic elements” earlier in the claim. In doing so, the District Court reached its erroneous conclusion that the claim necessarily requires “multiple logic elements.”

VII. ARGUMENT

A. Standard of Review

Where, as here, the District Court reviews only evidence intrinsic to the patent (the patent claims, specification, and prosecution history), the District Court's determination is reviewed *de novo*. See *Teva Pharma. USA, Inc. v. Sandoz, Inc.*, 574 U.S. ____ (2015); *Markman v. Westview Instruments, Inc.*, 517 U.S. 370 (1996); *Cybor Corp. v. FAS Techs., Inc.*, 138 F.3d 1448, 1454 (Fed. Cir. 1998)(*en banc*).

B. Claim 1 Should Properly Be Construed To Recite “*One Or More*” Logic Elements

1. “Logic Element” Is Introduced As Singular Or Plural

Claim 1 of the ‘422 Patent plainly recites “one nondeterministic logic element” when that term is first introduced in the first element of the body of Claim 1 (referred to herein as “location (a)”). So it cannot be disputed that at least at location (a), Claim 1 envisions a single logic element.¹

Looking further, the first element of Claim 1 recites that there is “one logic element” for each of “one or more variables.” See Claim 1,

¹For simplicity of discussion only, the phrase “logic element” will be used in place of “nondeterministic logic element.” None of the issues before this Court turn on the use or meaning of the word “nondeterministic.”

location (a). Simple logic dictates that if there is “one logic element” for each of “one or more variables,” then there must be “one or more logic elements.” In other words, if there is “one variable” then there could be “one logic element.” It is undisputed that this is the only reasonable interpretation of the language of Claim 1, at least at location (a).

Accordingly, the only reasonable meaning of “one logic element for each of one or more variables” is “one or more logic elements.”

2. Latter Uses Of “Logic Element” Necessarily Refer To Initial Use Of “Logic Element”

As the Court is aware, use of the word “the” before a noun in patent claims essentially requires that there be antecedent basis for that noun. See, e.g., *Energizer Holdings Inc. v. Int’l Trade Comm’n*, 435 F.3d 1366 (Fed. Cir. 2006)(discussing requirement of antecedent basis). Accordingly, when the word “the” precedes a claim term, and that claim term is introduced earlier in the claim, then the latter instance of the claim term necessarily refers to the earlier instance of the claim term. See *Papyrus Technology Corp. v. New York Stock Exchange, Inc.*, 581 F.Supp.2d 502, 533 (S.D.N.Y. 2008)(“To construe the plain meaning of the language at issue, the court must determine the antecedent basis of the word ‘said’ in the phrase.”); *Illinois Computer Research LLC v.*

Harpercollins Publishers, Inc., Case No. 10 Civ. 9124 (S.D.N.Y. January 18, 2012)(“[T]he law presumes that the use of the word ‘said’ in a claim is referring to an earlier use of that term within the patent.”).

In the instant case, when the term “logic element” appears in the latter part of Claim 1, the word “the” precedes the words “logic elements.” See Claim 1, “location (b).” Accordingly, based on ordinary and established principles of claim drafting, the term “the logic elements” at location (b) must refer to the initial term “logic element” at location (a), i.e., the one understood to be “one or more.”

3. Latter Uses Of Same Term Maintain Same Plural Nature Of Initial Claim Term

When the term “logic elements” appears at location (b) of the claim, the word “elements” is plural. However, that is unsurprising because, as just established, “**the** logic elements” at location (b) necessarily refers to the “**one or more** logic elements” that were introduced at location (a). See *supra*, Section B.1. Proper grammar requires that the term “logic elements” be plural since it follows the compound adjective “one or more,” which is both singular and plural. In other words, “one or more logic elements” (plural) is correct; “one or more logic element” (singular) is incorrect.

Accordingly, at location (b) the term “the logic elements” must be plural because it refers to its antecedent basis, which can only mean “one or more logic elements.” Where a claim term is introduced as singular or plural, latter references to that term maintain the same singular or plural nature. See *Baldwin Graphic Systems, Inc. v. Siebert*, 512 F.3d 1338, 1342-43 (Fed. Cir. 2008)(subsequent use of definite articles “the” or “said” in a claim to refer back to the same claim term does not change the general plural rule).

The use of the word “all” in conjunction with “the logic elements” does not change this result. In other words, there is nothing grammatically wrong or improper about the phrase “all of the one or more logic elements.” In short, the word “all” merely requires that “all” of the logic elements ***which exist*** have a common synchronization means. It does not require that more than one exist. So if only one logic element exists, then only one logic element is coupled to the synchronization means. Nothing about the plain language of the claim contravenes this interpretation.

C. The Claim Construction of “Logic Elements”

Although several terms were construed by the District Court, only its construction of “logic elements” is on appeal. What follows is a synopsis of the District Court’s claim construction of this term, and a discussion of the errors made by the District Court in its analysis.

1. The Competing Claim Constructions

Before the District Court were competing constructions for the term “coupled to all of the nondeterministic logic elements” from Claim 1 (see location (b)). Dr. Hangartner proposed that the term means “coupled to all of the *one or more* nondeterministic logic elements,” while Intel countered that the term means “coupled to all of the *multiple* nondeterministic logic elements.” The parties submitted opening and responsive briefs setting out their respective positions. (A078-164, A165-196, A325-362, A280-298). The District Court conducted a claim construction hearing at which the parties presented their positions.

After the hearing, the District Court issued its claim construction order agreeing with Intel and finding that “coupled to all of the

nondeterministic logic elements” means “coupled to all of the *multiple* nondeterministic logic elements.” (A003).

2. The District Court’s Flawed Linguistic Claim Construction Analysis

The District Court’s claim construction analysis is flawed as being primarily based on the reasoning that the claim “describes a single logic element in paragraph 1 (‘one nondeterministic logic element’) but pluralizes ‘logic elements’ twice in paragraph 3.” (A009). The District Court’s fundamental analysis is entirely linguistic, and relies on the plural nature of “logic elements” in location (b) of the claim, even though the term refers back to the “one or more logic elements” introduced at location (a) of the claim.

The District Court even acknowledged that the claim language clearly envisions a circuit having “one or more” logic elements. (A0010) (“[t]he language in preceding paragraphs seems to contemplate a logic circuit that could consist of a single logic element”). However, the District Court erred by failing to appreciate that the term “logic elements” at location (b) of the claim is only plural because it refers necessarily to the “one or more logic elements” that form its antecedent basis; it does not alter the earlier singular *or* plural meaning. See

Baldwin Graphic Systems, Inc. v. Siebert, 512 F.3d 1338, 1342-43 (Fed. Cir. 2008)(“Because the initial indefinite article (‘a’) carries either a singular or plural meaning, any later reference to that same claim element merely reflects the same potential plurality.”).

The District Court’s reliance on the plural nature of “the logic elements” when referring to the earlier instance of “one or more logic elements” constitutes reversible error in the District Court’s claim construction.

3. The District Court’s Flawed Prosecution History Analysis

In its analysis, the District Court also pointed to the prosecution history of the ‘422 Patent as being merely “consistent” with, but “not dispositive” of, its finding. (A009-010). In doing so, the District Court noted that Dr. Hangartner added the final clause of Claim 1 during prosecution. However, deviating from the conventional rule that “one or more” necessarily encompasses the singular requires significantly more showing than a prosecution history which is merely “consistent with” such a finding. Indeed, an applicant must evince a “clear intent” to deviate before the ordinary meaning of “one or more” can be abrogated. See *Baldwin Graphic Systems, Inc.*, 512 F.3d at 1342.

The passage of the prosecution history cited by the District Court does nothing to suggest that “more than one” logic element is required by the claim. Rather, the cited passage simply refers to a “synchronization means” which is used to keep however many logic elements as may be in the circuit operating properly. Nothing about the “synchronization means” compels the conclusion that more than one “logic element” is necessary.

More specifically, the District Court found that the “synchronization means” is composed of “DDelay” and “signal 32” disclosed in the ‘422 Patent specification. (A014-015). However, and referring to Fig. 6 of the ‘422 Patent, “DDelay” generates “signal 32” which generates a number of control signals for the components of the individual logic element depicted in Fig. 5. (A036). Those control signals “synchronize the operation” of each of the logic elements. For example, signal QLD directly controls operation of power transistor M18, and signals QNDEQ and /QNDEQ directly control operation of charge-equalizing transistors M16/M17. (A036). Proper synchronization of those elements is critical to the operation of even *one* logic element exactly as recited in the claim. In other words, if signal QLD is not

synchronized with signals QNDEQ and /QNDEQ, then even one logic element will not operate properly.

Thus, nothing about the file history evidences a “clear intent” to transform “one or more” logic elements into “two or more” logic elements. The District Court’s prosecution history analysis is a non-sequitur; a term being added during prosecution does not necessarily alter the meaning of preexisting claim terms.

4. The District Court’s Flawed Reliance On The Problem Solved By The Inventions Of Claims 2-10

The final error made by the District Court was its reliance on the argument that, essentially, Claim 1 is directed at solving a complex computing problem, and one logic element would be insufficient to solve such a complex computing problem. (A011-012). However, the District Court’s analysis is fatally flawed. Fundamentally, the District Court’s analysis improperly injects a new limitation into the claim based solely on the description of the preferred embodiment. See *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 906 (Fed. Cir. 2004) (“[T]his court has expressly rejected the contention that if a patent describes only a single embodiment, the claims of the patent must be construed as being limited to that embodiment”). But more substantively, the District

Court's analysis is based on a misunderstanding of the interplay between Claim 1 and the other claims of the '422 Patent.

The entire disclosure of the '422 Patent (not just Claim 1) is directed at solving complex computing problems, such as the NP complete problem. (A082-083). However, each different claim of a patent is directed to a different invention, and not necessarily all address the same problem. See, e.g., *Environmental Designs, Ltd. v. Union Oil Co.*, 713 F.2d 693, 699 (Fed. Cir. 1983). Such is exactly the case here with the '422 Patent.

More specifically, the inventions recited in claims 2-10 can be considered a machine for solving a very difficult computer problem by using random numbers as "guesses." However, the invention recited in claim 1 can be considered a machine for solving a related but slightly different computer problem; the need for random numbers. Stated simply, claim 1 builds a special tool while claims 2-10 use that tool to solve a problem.

The District Court erroneously relied on the difficulty of the larger problem attacked by the full-system claims (Claims 2-10) to improperly inject a new limitation into the narrower sub-system claim (Claim 1).

Indeed, the District Court’s reasoning is in direct contrast with the plain language of the claim. More specifically, the District Court reasoned as follows:

By using multiple logic elements at once, the user can generate many strings of random numbers simultaneously. This is much more efficient than relying on a single element to produce random numbers one at a time. . . . Thus it appears that the inefficient solution—that is, patenting circuits containing only one logic element—is inconsistent with both the proposed solution to the complex computing problems at hand and the preferred embodiment of the invention.”

(A012).

However, the plain language of the claim recites that the problem is expressed as “one or more clauses” in “one or more variables.” The District Court’s analysis completely ignores that the plain language of the claim **explicitly** envisions a single-clause, single-variable problem. A claim construction that is in disagreement with the plain language of the claim is rarely if ever correct. See *Neomagic Corp. v. Trident Microsystems, Inc.*, 287 F.3d 1062, 1075 (Fed. Cir. 2002).

The only way the “single clause, single variable” claim language has any meaning at all is if the entire claim envisions “one or more logic elements.” Accordingly, the District Court’s reasoning that a proper

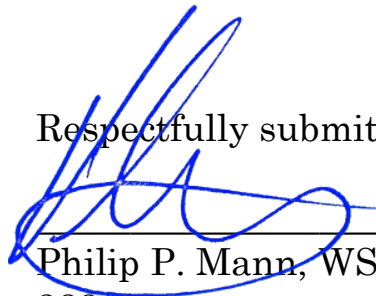
construction must ***exclude*** the “single logic element” scenario is plainly at odds with the language of the claim, and should be vacated.

VIII. CONCLUSION

The District Court improperly concluded that the term “one or more logic elements” actually means “multiple logic elements.” The plain language of the claim and the ‘422 Patent reveal that a minimum of one logic element is required, not two. Accordingly, Dr. Hangartner respectfully requests that the District Court’s claim construction order be vacated as to that finding.

Dated: April 30, 2015

Respectfully submitted



Philip P. Mann, WSBA No:
28860

MANN LAW GROUP

John Whitaker, WSBA No:
28868

WHITAKER LAW GROUP

1218 Third Avenue, Suite 1809
Seattle, Washington 98101
(206) 436-0900

phil@mannlawgroup.com

john@wlawgrp.com

Attorney for Plaintiff/Appellant
Dr. Ricky Hangartner

ADDENDUM

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF OREGON
PORTLAND DIVISION

RICKY D. HANGARTNER,

Plaintiff,

v.

INTEL CORPORATION,

Defendant.

MOSMAN, J.,

No. 3:14-cv-00141-MO

OPINION AND ORDER

Plaintiff Ricky Hangartner brings this action against Defendant Intel Corporation alleging infringement of United States Patent No. 6,463, 422 (the ‘422 Patent). Dr. Hangartner alleges Intel sells processors that incorporate circuitry and techniques disclosed in Claim 1. Intel denies infringement and counterclaims that the ‘422 Patent is invalid.

Dr. Hangartner and Intel dispute the meaning of nine phrases in Claim 1. The parties filed initial and responsive claim construction memoranda and the court held a claim construction hearing on November 17, 2014. Based on the parties’ evidence, memoranda, and the argument of counsel, I issued a brief Opinion and Order setting forth my construction of the nine disputed phrases on November 20, 2014. This explanatory opinion discusses how I applied the governing claim construction standards to the phrases to arrive at these constructions.

CLAIM CONSTRUCTION STANDARDS

The court is charged with determining the meaning of ambiguous claim language as a matter of law. *Markman v. Westview Instruments, Inc.*, 517 U.S. 370, 372 (1996). (“[T]he

construction of a patent, including terms of art within its claim, is exclusively within the province of the court.”). When an ambiguity arises, the court must assign the term the meaning that it would have “to a person of ordinary skill in the art in question at the time of the invention.” *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005). This approach is intended to create an “objective baseline from which to begin claim interpretation.” *Id.* at 1313. “In the end, the court’s ultimate goal is to construe the disputed terms in a manner consistent with the way the inventor defined them and a person of ordinary skill in the art would understand them.” *Skedco, Inc. v. Strategic Operations, Inc.*, No. 03:13-cv-00968-HZ, 2014 WL 4385752, at *5 (D. Or. Sept. 3, 2014).

Construing a disputed term as a person of ordinary skill in the art would understand it requires the court to review multiple sources of evidence, both intrinsic and extrinsic to the patent itself. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1586 (Fed. Cir. 1996). Intrinsic evidence encompasses the words of the claim themselves, the patent specification, and those portions of the patent prosecution history entered into evidence. *Id.* The court considers extrinsic evidence only when it is otherwise insufficient to resolve the ambiguity. *Id.* at 1583.

Some of these sources of evidence are “more valuable than others.” *Phillips*, 415 F.3d at 1324. The claim language itself contains the most valuable evidence of its own meaning. *Vitronics*, 90 F.3d at 1582 (“First, we look to the words of the claims themselves...to define the scope of the patented invention.”) There is a “heavy presumption” that these words carry their “ordinary meaning,” as observed “through the viewing glass of a person skilled in the art.” *Johnson Worldwide Assocs., Inc. v. Zebco Corp.*, 175 F.3d 985, 989 (Fed. Cir. 1999); *Ferguson-Beauregard/Logic Controls v. Mega Sys., LLC*, 350 F.3d 1327, 1338 (Fed. Cir. 2003).

Beyond the plain language of the claims, the patent specification is always “highly relevant” and often dispositive to the proper construction. *Vitronics*, 90 F.3d at 1582 (“[I]t is the

single best guide to the meaning of a disputed term.”). The purpose of the patent specification is to teach and enable those skilled in the art to make and use the invention, along with the best method for doing so. *Cyber Acoustics, LLC v. Belkin Int’l., Inc.*, No. 3:13-cv-01144-SI, 2014 WL 1225198 (D. Or. Mar. 24, 2014), quoting *Phillips*, 415 F.3d at 1323. The inventor can use the specification to describe the invention in a number of ways, such as describing different “embodiments” of the invention and by assigning particular meanings to specific claim language. *Metabolite Lab., Inc. v. Lab. Corp. of Am. Holdings*, 370 F.3d 1354, 1360 (Fed. Cir. 2004); *Phillips*, 415 F.3d at 1316. In the first instance, the embodiments serve as illustrative examples of the invention claimed. *Phillips*, 415 F.3d at 1323 (“One of the best ways to teach a person of ordinary skill in the art how to make and use the invention is to provide an example of how to practice the invention in a particular case.”). In the second instance, the inventor clarifies that he or she intends the claim language to carry a specific meaning in contravention to the meaning it would otherwise possess. *Id.* In these cases, “the inventor’s lexicography governs.” *Id.* at 1316.

Finally, the prosecution history, which contains the record of the proceedings before the Patent and Trademark Office, informs the analysis into what a person skilled in the art would understand the term to mean. *Vitronics*, 90 F.3d. at 1582–83. The prosecution history becomes useful where it “provides evidence of how the PTO and the inventor understood the patent.” *Phillips*, 415 F.3d at 1317. However, this evidence is less valuable in that it represents the “ongoing negotiation” between the inventor and the PTO. *Id.* The final result of that negotiation, the patent itself, provides better evidence of the claim’s intended meanings at the time the patent issued. *Id.*

Taken together, “[t]he claims, specification and file history, rather than extrinsic evidence, constitute the public record of the patentee’s claim, a record on which the public is

entitled to rely.” *Vitronics*, 90 F.3d.at 1583. This intrinsic evidence forms the basis of the claim construction analysis below. As I find it is sufficient to resolve the ambiguities in the claim language, I need not resort to the use of extrinsic evidence. *Id.*

DISCUSSION

I. Overview of the ‘422 Patent

The ‘422 Patent discloses a computing system that uses random numbers, or boolean values, to solve extremely difficult computing problems. The overall computing system is comprised of a number of subsystems whose functions are coordinated as follows: One subsystem generates random numbers that act as proposed solutions to the problem; another subsystem tests the random numbers against the problem; and another creates feedback as to the result of this test. The ‘422 Patent discloses this overall system in ten claims, but only the subsystem that generates the random numbers is at issue in this suit. This “nondeterministic” subsystem is described in Claim 1, which reads:

“A nondeterministic logic circuit for generating random boolean values of one or more variables as a proposed solution to a computing problem expressed in conjunctive normal form as one [or] more clauses in said one or more variables, the logic circuit comprising:

- one nondeterministic logic element for generating a respective random boolean value for each one of the said one or more variables; and
- each nondeterministic logic element comprising:
 - a cross coupled pair of transistor inverter circuits;
 - means for controlling power to the cross-coupled pair of transistor inverter circuits;
 - and
 - means for equalizing charge on the gates of the transistor inverter circuits while power is removed from the cross-coupled pair, thereby driving the cross coupled pair to an unstable equilibrium, whereby intrinsic circuit noise will cause the cross-coupled pair to randomly assume one of two stable states when power is restored to the cross coupled pair, the stable state assumed by the cross coupled pair providing a probabilistically selected random boolean value

and further comprising common synchronization means coupled to all of the nondeterministic logic elements for synchronizing operation of the nondeterministic logic elements.”

The preamble sets forth the type of invention claimed: here, a “nondeterministic logic circuit for generating random boolean values.” It ends by introducing a list of components that form the logic circuit (“the logic circuit comprising:”). The remainder of claim 1 consists of three paragraphs, each providing detailed information about the components of the logic circuit. The first paragraph following the colon, herein referred to as paragraph 1, indicates the logic circuit is comprised of “one nondeterministic logic element ... for each one of the said one or more variables” in the computing problem to be solved. The subsequent paragraph, herein referred to as paragraph 2, indicates each of the nondeterministic logic elements is itself comprised of three separate components. The last paragraph in the claim, herein referred to as paragraph 3, indicates the logic circuit is “further comprising” of a “common synchronization means coupled to all of the nondeterministic logic elements for synchronizing operation of the nondeterministic logic elements.” Thus at a high level, the claim indicates Dr. Hangartner’s invention consists of a logic circuit containing at least one logic element, itself comprised of three separate components, as well as a means of synchronizing each of the logic elements contained within the circuit.

II. Claim Construction

A. Minimum Number of Logic Elements in the Circuit

The primary issue of claim construction is whether claim 1 should be construed to require multiple logic elements within the circuit, or whether a single logic element suffices. Dr. Hangartner argues a logic circuit containing one logic element is sufficient. Intel disagrees, arguing only those circuits containing multiple logic elements fall within the scope of the claim. Both parties consider this claim construction issue to be dispositive of the underlying infringement dispute.

The claim construction analysis begins with the examination of the words of the claim themselves. *Vitronics*, 90 F.3d at 1582. Rather than reading each of the nine disputed phrases in isolation, reading the claim as a whole reveals an internal inconsistency that warrants further construction. Specifically, Dr. Hangartner’s use of singular nouns in paragraph 1 is in tension with his use of plural nouns in paragraph 3. That is, he describes a single logic element in paragraph 1 (“one nondeterministic logic element”) but pluralizes “logic elements” twice in paragraph 3 when describing the circuit’s synchronization means (“and further comprising common synchronization means coupled to all of the nondeterministic *logic elements* for synchronizing operation of the nondeterministic *logic elements*”) (emphasis added). Describing the logic elements both in the singular and in the plural creates uncertainty regarding the minimum number of logic elements the claim contemplates.

The parties offer different interpretations to explain this inconsistency. Dr. Hangartner argues the singular language in paragraph 1 indicates the minimum number of logic elements must be one. In contrast, Intel argues the plural language in paragraph 3 indicates the minimum number of logic elements must be greater than one. Therefore in Intel’s view, a logic circuit containing only one logic element, but no more, falls outside the scope of the claim.

I conclude that the plain meaning of the term “logic elements” requires multiple logic elements to be present within the circuit. First, I assume Dr. Hangartner, as the inventor and the author of the claim language, deliberately pluralized “logic elements” in paragraph 3. He specified that the synchronization means is coupled to “all” of the logic “elements” in order to synchronize the operation of the logic “elements.” Interpreting this to mean that only one logic element could be present would require “all” to mean “each,” and “elements” to mean “element.” This interpretation stretches his language to include the singular where he chose to use the plural;

this stretches the ordinary meaning too far. Had he intended circuits comprised of only one logic element to fall within the scope of his claim, use of the singular would have accomplished this goal. However, he chose to refer to elements in the plural, the ordinary meaning of which must be “more than one element.” Pluralizing “logic elements” in paragraph 3 indicates that his claim contemplates a circuit comprising, at a minimum, “more than one” logic element.

Notably, the language contemplating multiple logic elements does not appear until paragraph 3, where it refers to “common synchronization means.” The language in preceding paragraphs seems to contemplate a logic circuit that could consist of a single logic element, in addition to the common synchronization means described in paragraph 3. Specifically, paragraph 1 plainly refers to a logic circuit comprising “one nondeterministic logic element.” Just as it would stretch the language to read “elements” as “element” in paragraph 3, the converse is also true in paragraph 1: It would stretch the language too far to read “element” as “elements.” Only the plural language of paragraph 3 widens the purview of the claim to include, and in fact require, multiple logic elements to be present within the circuit.

This linguistic shift may be explained in part by the prosecution history of the ‘422 patent. In an earlier version of the patent application, Dr. Hangartner claimed a logic circuit that did not include a common synchronization means. *See* October 4, 2000 Response to Office Action [45-7] at 5–6. After the USPTO rejected the application, Dr. Hangartner added the synchronization requirement onto what is now claim 1. May 13, 1999 Amendment [45-2] at 3, 8 (amending the claim to include the limitations from cancelled claim 9, such as synchronization). As he explained the revised claim to the USPTO in his Opening Appeal Brief in 2001:

[t]his circuit claim includes essentially two elements: (a) one nondeterministic logic element for each variable, to generate a random boolean value for the corresponding variable; and (b) a “common synchronization means” that synchronizes operation of the nondeterministic logic elements. They have to be

synchronized because each guess at a solution to the problem requires that a random value be picked for every variable in the problem.

[45-3] at 5. This indicates that Dr. Hangartner argued on appeal that his logic circuit contains two components, a logic element for each variable plus a common synchronization means, whereas originally he had argued that the logic circuit was patentable without the synchronization means. The fact that Dr. Hangartner “tacked on” the synchronization requirement to the preexisting, yet apparently unpatentable, claim might explain the sudden shift from the singular reference to “one nondeterministic logic element” in paragraph 1 to the plural references to “logic elements” in what now appears as the synchronization requirement in paragraph 3.

To be clear, this is not to say that this evidence from the prosecution history is dispositive. Rather, I find the prosecution history is consistent with the construction required from the plain meaning of the term “logic elements” in paragraph 3. The prosecution history here further “inform[s] the meaning of the claim language by demonstrating how the inventor understood the invention[.]” *Phillips*, 415 F.3d at 1317. Here, it indicates that Dr. Hangartner understood the final invention to include both one logic element for each variable as well as a means of synchronizing the operation of multiple logic elements within the circuit. His explanation of the invention in the prosecution history sheds light on the internal inconsistency between the references to a single “logic element” in paragraph 1 and the reference to “logic elements” in paragraph 3 that is otherwise resolved on a linguistic basis.

Moreover, requiring multiple logic elements within the circuit is also consistent with Dr. Hangartner’s proposed solution to the specific complex computing problems at hand. The background of the invention explains that “Nondeterministic Polynomial Time Complete (NP-complete) decision problems are those for which no efficient solution method is known[.]” ‘422

Patent 1:14–16. These problems are so complex that one way to solve them is simply to guess at the answer by using random numbers, and then check to see which of the many guesses was in fact correct. The parties agree that claim 1 describes a random number generator, structured as a logic circuit. They also agree that within the circuit, an individual logic element can generate its own string of random numbers, one after the other. *See, e.g.*, ‘422 Patent 7:3–5 (“[e]ach individual ND element...generates one of the probabilistic variables and its complement”). Furthermore, they agree that the circuit can be “scaled up” to include multiple logic elements. *See, e.g.*, ‘422 Patent 8:11–12 (“in an integrated circuit, individual ND elements could be grouped, for example in groups of 8 or 16 such elements”). By using multiple logic elements at once, the user can generate many strings of random numbers simultaneously. This is much more efficient than relying on a single element to produce random numbers one at a time.

Dr. Hangartner’s claimed invention, consisting of all ten of his claims, is clearly aimed at producing efficient solutions to these difficult problems, not inefficient ones. The first line of the ‘422 Patent abstract describes the invention as a computing system that “provides computational functionality needed to *efficiently* realize randomized computing methods in otherwise standard, deterministic digital computing systems” (emphasis added). Also, the written description of the preferred embodiment specifically contemplates adding multiple logic elements to the circuit: “[i]n a practical implementation in an integrated circuit, individual [nondeterministic logic] elements could be grouped, for example, in groups of 8 or 16 such elements[.]” ‘422 Patent 8:11–13. Thus it appears that the inefficient solution—that is, patenting circuits containing only one logic element—is inconsistent with both the proposed solution to the complex computing problems at hand and the preferred embodiment of the invention. And while the abstract and written description do not delimit the claim language *per se*, “[s]uch intrinsic evidence is the

most significant source of the legally operative meaning of disputed claim language.” *Vitronics*, 90 F.3d at 1582.

Read in light of the plain meaning of “logic elements” and “synchronization” in paragraph 3, it becomes clear that Dr. Hangartner’s claim can only encompass those logic circuits containing more than one logic element; a minimum of two. This conclusion is consistent with the intrinsic evidence from the patent’s abstract, written description, and prosecution history. Therefore I construe the term “coupled to all of the nondeterministic logic elements” as “coupled to all of the *multiple* nondeterministic logic elements.”

B. Conjunctive Normal Form

The preamble states Dr. Hangartner’s logic circuit presents “a proposed solution to a computing problem,” specifically, a problem which is “expressed in conjunctive normal form as one [or] [sic] more clauses in said one or more variables[.]” The parties agree that “conjunctive normal form” is an ambiguous term requiring further construction. Each proposes a construction drawn from definitions of the term contained in the patent’s written description. Intel cites to the definition found at 6:30–33: “a series of clauses logically ANDed together, each of the clauses consisting of a series of variables, or literals, logically ORed together.” Dr. Hangartner cites to a very similar definition found at 4:4–6: “a series of clauses, each clause made up of literals (variables or their complements)[.]” The key distinction between the two is that Intel’s definition suggests that “conjunctive normal form” describes only those problems containing multiple variables, whereas Dr. Hangartner’s suggests problems containing a single variable can be correctly expressed in “conjunctive normal form.” By describing the computing problem as necessarily containing a “series of variables,” Intel argues computing problems containing a single variable fall outside the scope of the claim.

Here, the plain language of the preamble is dispositive. The terms “one [or] more clauses” and “one or more variables” clearly indicates a computing problem containing a single clause and a single variable would fall within the scope of the claim. This construction does not limit the claimed circuit to solving single variable problems; rather, it simply means that multivariable problems are not required. And while Intel correctly points out that an inventor’s own definition is controlling, this is a case where the inventor has proffered multiple definitions for the same term. *See C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 862 (Fed. Cir. 2004). Where one definition is more consistent with the plain language of the claim, the plain language must decide the issue. *See Vitronics*, 90 F.3d at 1582 (“First, we look to the words of the claims themselves...to define the scope of the patented invention.”). Thus I adopt Dr. Hangartner’s construction of “conjunctive normal form” “described as a series of one or more clauses, each clause made up of one or more literals (variables or their complements).”

C. Synchronization

“Synchronization” appears within a “means-plus-function” limitation under 35 U.S.C. § 112 ¶ 6 (“common synchronization means coupled to all of the nondeterministic logic elements for synchronizing operation of the nondeterministic logic elements”). A “means-plus-function” limitation is essentially a form of shorthand that allows Dr. Hangartner to import a structural limitation from the patent’s written description into the claim without having to describe it in detail within the claim itself. *See, e.g., Enviroco Corp. v. Clestra Cleanroom, Inc.*, 209 F.3d 1360, 1364 (Fed. Cir. 2000). The parties agree that the structure must include signal 32 and delay element 64. Their proposed constructions differ only as to whether “inverters 80,” depicted in Figure 4 of the ‘422 Patent, is part of the structure that performs the function.

Intel argues that “inverters 80” is essential to the synchronization function because it “fans out” synchronization signals among multiple logic elements. The significance of this argument seems to be that if “inverters 80” functions as Intel claims, that fact provides further evidence that synchronization can only occur in circuits containing multiple logic elements. Dr. Hangartner disagrees, arguing “inverters 80’s” plays only a simplistic, nonessential role during synchronization.

Whether “inverters 80” or its functional equivalent plays an important role in synchronization is inapposite to the conclusion that synchronization must occur within the context of multiple logic elements, for the reasons stated above. This conclusion moots Intel’s argument that “inverters 80” is essential to synchronization. Absent additional evidence that “inverters 80” is in fact critical to synchronization, I decline to read Intel’s proposed limitation into the claim. *See SciMed Life Sys. v. Advanced Cardiovascular Sys., Inc.*, 242 F.3d 1337, 1340 (Fed. Cir. 2001) (stating “one of the cardinal sins of patent law [is] reading a limitation from the written description into the claims”). Therefore I adopt Dr. Hangartner’s construction as follows: The phrase “common synchronization means coupled to all of the nondeterministic logic elements for synchronizing operation of the nondeterministic logic elements” has the function of “synchronizing operation of the nondeterministic logic elements,” and its corresponding structure is “signal 32, and delay element 64.”

D. Correlation between Logic Elements and Variables

As is true of the synchronization dispute described above, requiring multiple logic elements moots the issue of whether the preamble requires a “one-to-one” or “one-to-many” correlation between the logic elements in the circuit and the variables in the computing problem. The thrust of Intel’s argument is that because the preamble suggests the circuit can solve

multivariable problems, and one logic element can pair with only one variable, the logic circuit must contain multiple logic elements (an exclusively “one-to-one” correlation). Dr. Hangartner disagrees, citing to the written description’s statement that “one of the individual ND logic elements ... can be configured to generate random values for all of the variables” (a non-exclusive “one-to-many” correlation). ‘422 Patent 7:14, 24–25. Dr. Hangartner also asserts the language in question, “one nondeterministic logic element for generating a respective random boolean value for each one of the said one or more variables,” unambiguously suggests that both one-to-one and a one-to-many correlations are possible.

I agree with Dr. Hangartner that the term is unambiguous. The terms “one logic element” and “one or more variables” plainly suggest that one element could pair with one variable or with more than one. On a broader level, the multiple logic element requirement that Intel suggests appears in the preamble, in fact appears only in paragraph 3 in the context of synchronization. Thus I decline to adopt Intel’s proposed construction that would limit the claim to “one-to-one” correlations and find no further construction of the term is necessary.

E. Unstable Equilibrium

Paragraph 2 indicates that each logic element contains three components: a pair of inverters cross-coupled together, a means for controlling power to the pair, and a means for equalizing its electrical charge under particular circumstances; namely, once the pair reaches a state of “unstable equilibrium.” The parties agree that this ambiguous term describes a condition in which a pair of inverters, cross-coupled together such that the output of one is typically the complement of the other, is forced into an unstable state where the output of each is now the same as the other. The parties further agree that even a small disturbance, such as thermal or “random” noise, could cause the pair of inverters to “fall out” of this unstable state and revert to

their stable, complementary state. The parties disagree as to whether the inverters' electrical charges can fluctuate while they exist in this forced, unstable condition.

Intel argues that one skilled in the art understands this unstable state to encompass a moment, however brief, in which the outputs of each inverter cease rising and falling and come to rest in equivalence with each other. At this moment, the output of each inverter, measured as electrical voltage, is unchanging. Dr. Hangartner argues that this moment may occur during "unstable equilibrium," or it may not. He believes it is possible that the outputs of each inverter never truly come to rest at the same, unchanging value because minute voltage fluctuations could continue to recur during this period of instability.

Intel replies that this unchanging state is better known in the industry as a "metastable state." Intel and Dr. Hangartner agree that a metastable state is a condition that occurs while power is being added to a typical circuit. However, Dr. Hangartner's logic circuit functions in a fundamentally different manner in that the equilibrium reached, however it is described, occurs only "while power is removed from the cross-coupled pair." While it is conceivable that the unstable equilibrium that occurs while power is removed from Dr. Hangartner's atypical circuit resembles the metastable state that occurs while power is added to a typical circuit, I find the claim language requires no such limitation. Thus I construe "unstable equilibrium" as "a condition where the inputs and outputs of a cross-coupled pair of inverters are in substantially the same state and are substantially unchanging and a small disturbance would produce a change away from that state."

The particular manner in which this pair of inverters functions is not readily apparent from the plain language of the claim, despite Dr. Hangartner's contention. Accordingly, I agree with Intel that the phrase describing the circuit's unique power off/power on mechanism requires

construction: “while power is removed from the cross-coupled pair, thereby driving the cross-coupled pair to an unstable equilibrium ... whereby the cross-coupled pair[] randomly assume[s] one of two stable states when power is restored to the cross-coupled pair.” Intel suggests the phrase should be construed as “the cross-coupled pair of inverters is driven to an unstable equilibrium while power is removed from the pair; when power is restored, the pair transitions directly from the existing unstable equilibrium state to a randomly assumed stable state.” This construction more clearly delineates the events occurring while power is removed from those events occurring while power is restored. However, describing the transition from the unstable state to the stable state as occurring “directly” is overly precise, particularly given the fact that the triggering event for this transition is inherent randomness. Thus I adopt Intel’s construction of this term with the word “directly” omitted.

F. Unambiguous Terms

The parties contend the phrase “equalizing charge on the gates of the transistor inverter circuits” in paragraph 2 requires additional construction. I disagree. This is evident from the fact that the parties’ proposed constructions differ only as to whether the means for “equalizing charge” should be described as reaching “the same charge” or “substantially the same charge.” However, the plain meaning of “equalizing” clearly invokes the concept of being “the same.” A construction requiring the charges to become either precisely or only substantially “the same” risks reading too much into language that is otherwise understood without confusion. *Phillips*, 415 F.3d at 1312.

The same is true for the phrase “cause the cross-coupled pair to randomly assume one of two stable states.” I agree with Dr. Hangartner that this phrase is unambiguous and should not be construed. He argues in the alternate that the court should adopt the parties’ identical

proposed construction, along with his qualification that “random” is described as “substantially random.” I decline to draw such a distinction for the same reasons described above.

Finally, the parties propose competing constructions of the term “proposed solution to a computing problem” in the preamble. Dr. Hangartner argues this should be rephrased as “a proposed solution to a problem that is solved with a computer.” Intel’s construction is more specific: a “set of values proposed to satisfy a NP-complete problem.” However, the plain meaning of the terms “proposed solution” and “computing problem” is readily apparent, making further construction unnecessary.

CONCLUSION

The Court construes the disputed ‘422 Patent claim terms as stated herein.

IT IS SO ORDERED.

DATED this 17th day of December, 2014.

/s/ Michael W. Mosman
MICHAEL W. MOSMAN
United States District Judge

(12) **United States Patent**
Hangartner

(10) **Patent No.:** **US 6,463,422 B1**
(45) **Date of Patent:** ***Oct. 8, 2002**

(54) **PROBABILISTIC COMPUTING METHODS AND APPARATUS**

(76) Inventor: **Ricky D. Hangartner**, 725 NW. Flanders #206, Portland, OR (US) 97209

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/898,528**

(22) Filed: **Jul. 22, 1997**

Related U.S. Application Data

(63) Continuation of application No. 08/296,654, filed on Aug. 26, 1994, now Pat. No. 5,680,518.

(51) **Int. Cl.**⁷ **G06N 3/02**

(52) **U.S. Cl.** **706/14; 706/52; 706/45**

(58) **Field of Search** **706/13, 52, 45**

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,812,473	*	5/1974	Tucker	711/151
5,148,513	*	9/1992	Koza et al.	706/13
5,390,282	*	2/1995	Koza et al.	706/13
5,680,518	*	10/1997	Hangartner	706/52
5,742,738	*	4/1998	Koza et al.	706/13
5,812,740	*	9/1998	Scepanovic et al.	706/45
5,835,381	*	11/1998	Scepanovic et al.	364/491
5,844,811	*	12/1998	Scepanovic et al.	364/491

OTHER PUBLICATIONS

Broder, et al, On the Satisfiability of andom 3-CNF Formulas, Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, 1993, pp. 322-330, Jan. 1993.*

Di Giacomo, Joseph, VLSI Handbook, McGraw-Hill Publishing Company, 1989, pp. 4.1-4.9, Jan. 1989.*

Horowitz et al, The Art Of Electronics, Cambridge University Press, 1989, p. 507, Jan. 1989.*

Millman, Jacob, Microelectronics: Digital and Analog Circuits and Systems, McGraw-Hill Publishing Company, 1979, pp. 282-296, Jan. 1979.*

Broder, Andrei Z., et al., On the Satisfiability of Random 3-CNF Formulas, Proceedings, of the ACM-SIAM Symposium on Discrete Algorithms, 1993, pp. 322-330, Jan. 1993.*

Millman, Jacob, MICROELECTRONICS: Digital and Analog Circuits and Systems, 1979, pp. 205-206, Jan. 1979.*

Broder, A.Z.; Dolev, D., Flipping Coins In Many Pockets (Byzantine Agreement On Uniformly Random Values), Foundations of Computer Science, 1984. 25th Annual Symposium on, pp.: 157-170, Jan. 1984.*

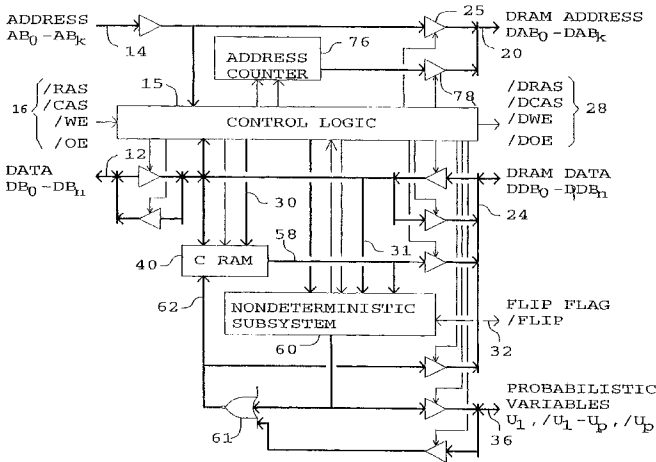
(List continued on next page.)

Primary Examiner—Kevin J. Teska
Assistant Examiner—Wilber Starks
(74) *Attorney, Agent, or Firm*—Stoel Rives LLP

(57) **ABSTRACT**

A new probabilistic computing system (PCS) provides computational functionality needed to efficiently realize randomized computing methods in otherwise standard, deterministic digital computing systems. The PCS may be incorporated in a standard computing platform such as a PC or workstation. In the PCS, a computational path includes a random access memory (RAM) where a predetermined computing problem is stored in conjunctive normal form. A nondeterministic subsystem generates random binary values forming a proposed solution to the problem, which solution is rapidly checked through a crosspoint switch array coupled to the RAM. The computational path essentially runs asynchronously, while a delay circuit provides delay and timing signals for interfacing with external DRAM, as well as a synchronizing signal for operation of several of the PCS systems together for enhanced performance.

10 Claims, 12 Drawing Sheets



US 6,463,422 B1

Page 2

OTHER PUBLICATIONS

Brown, S.; Rose, J., FPGA and CPLD architecture: a tutorial, IEEE Design & Test of Computers Volume: 13 2, Summer 1996, pp.: 42-57, Jan. 1996.*

Richard M. Karp and Avi Wigderson, A fast parallel algorithm for the maximal independent set problem; J. ACM 32, 4 (Oct. 1985), pp. 762-773.

Krishnamurthy, B.; Tollis, I.G., Improved techniques for estimating signal probabilities, Computers, IEEE Transactions on, vol. 38 7, Jul. 1989, pp.: 1041-1045.

Di Giacomo, Joseph, VLSI Handbook, McGraw Hill Publishing Company, 1989, pp. 4.1-4.9, Jan. 1989.

Signetics Corporation, Signetics Data Manuel, 1976, pp. 57-62, Jan. 1976.

Hamacher, V. Carl, Computer Organization, Second Edition, McGraw-Hill Publishing Company, 1978, pp. 537-538, Jan. 1978.*

Gear, C. William, Introduction to Computers, Structured Programming, and Applications, Module A: Applications and Algorithms in Science and Engineering, Science Research Associates, Inc., A Subsidiary of IBM, 1978, pp. A55, Jan. 1978.*

Koza, John R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, Massachusetts, Jan. 1, 1992.*

Millan, Jacob, "Microelectronics: Digital and Analog Circuits and Systems", McGraw Hill Book Company, 1979, pp. 208-209, Jan. 1979.*

Mano, Morris, "Computer System Architecture, Second Edition", Prentice Hall, p. 7, Jan. 1982.*

* cited by examiner

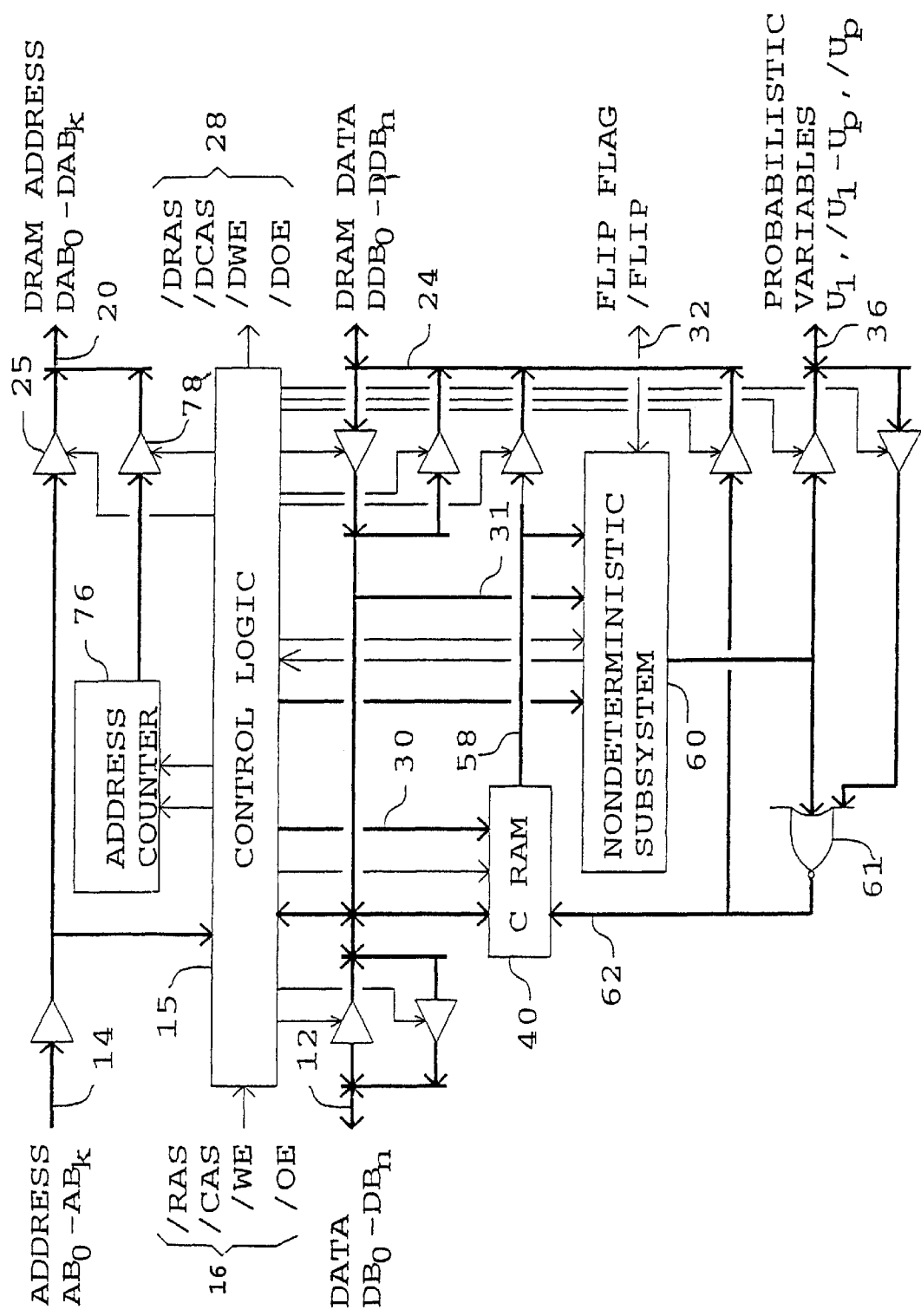


FIGURE 1

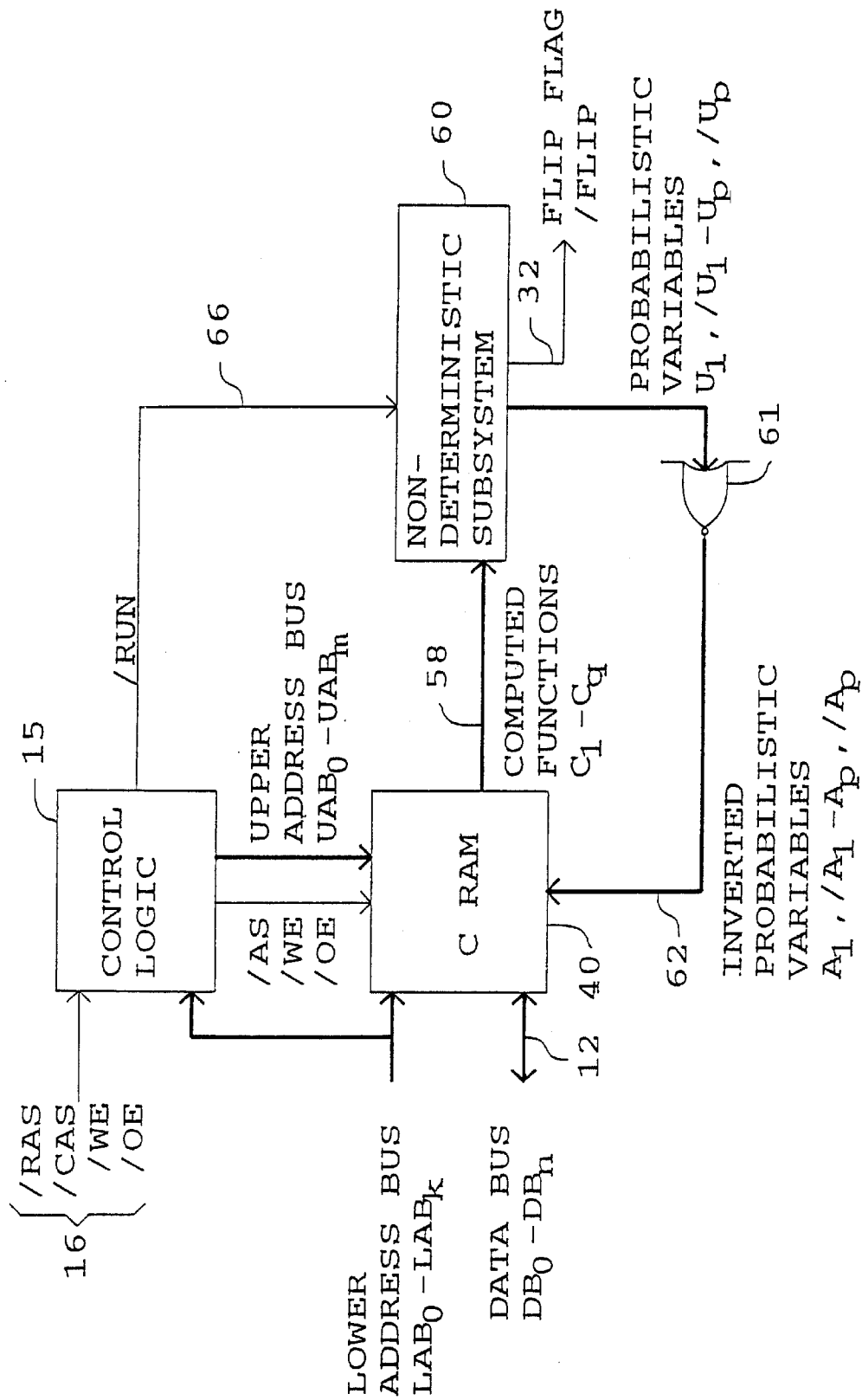


FIGURE 2

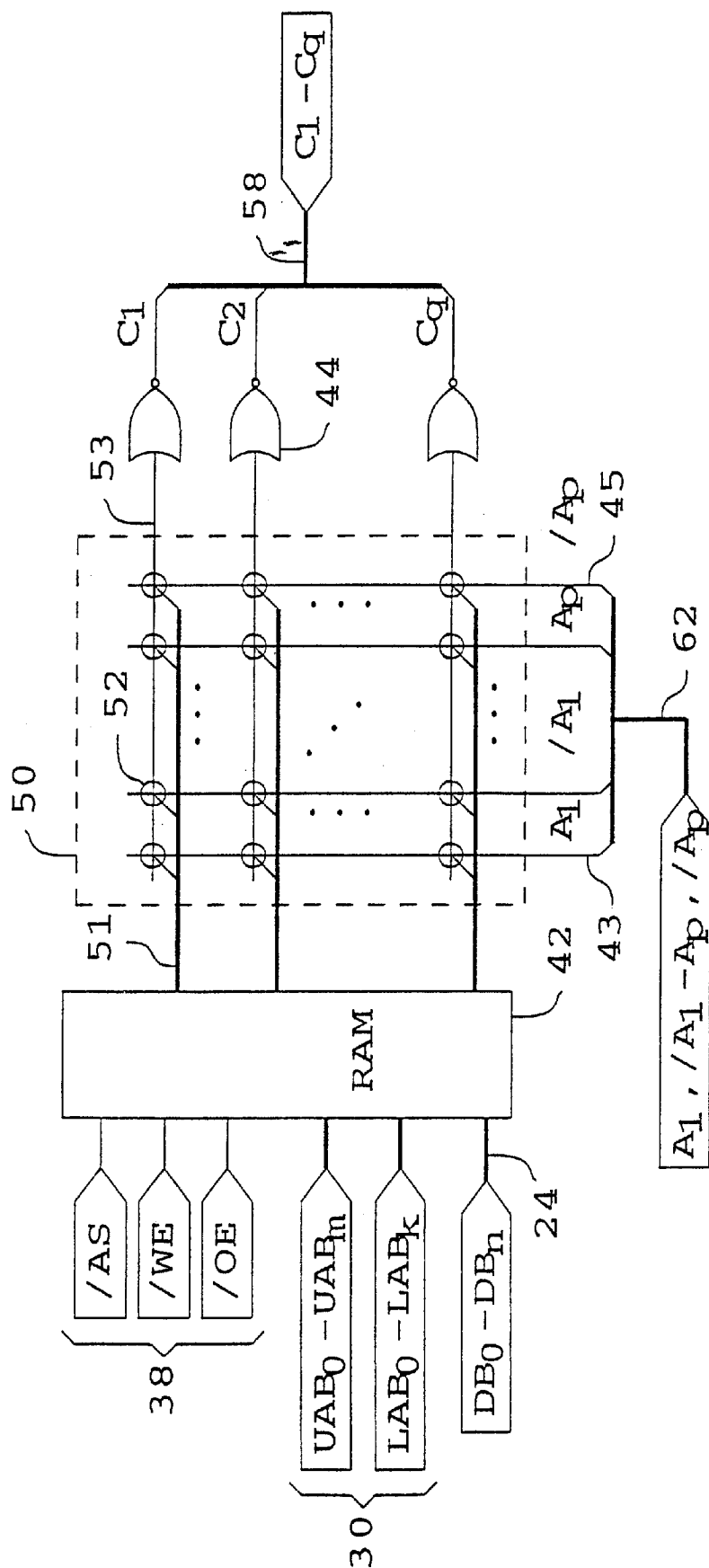


FIGURE 3

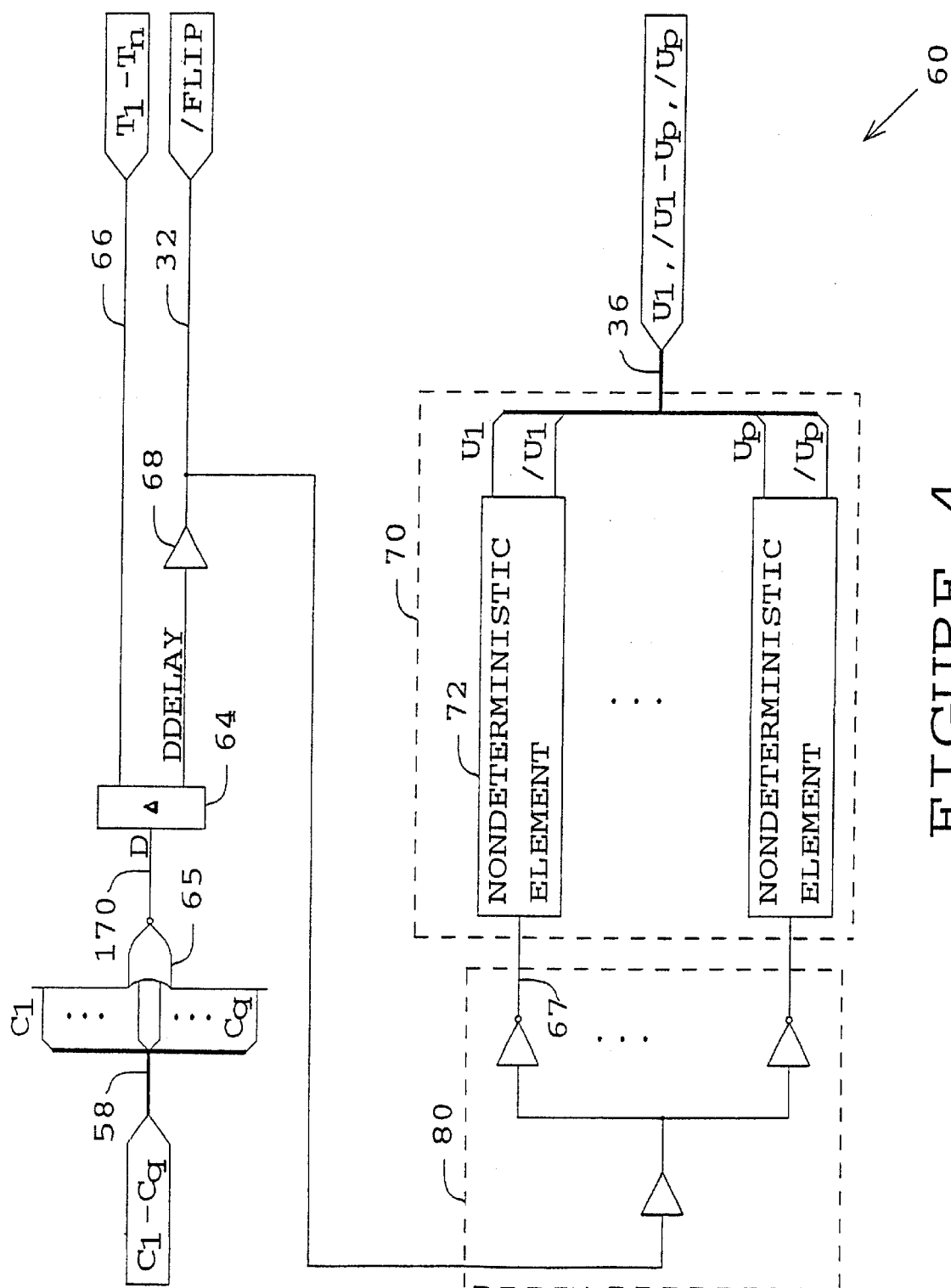


FIGURE 4

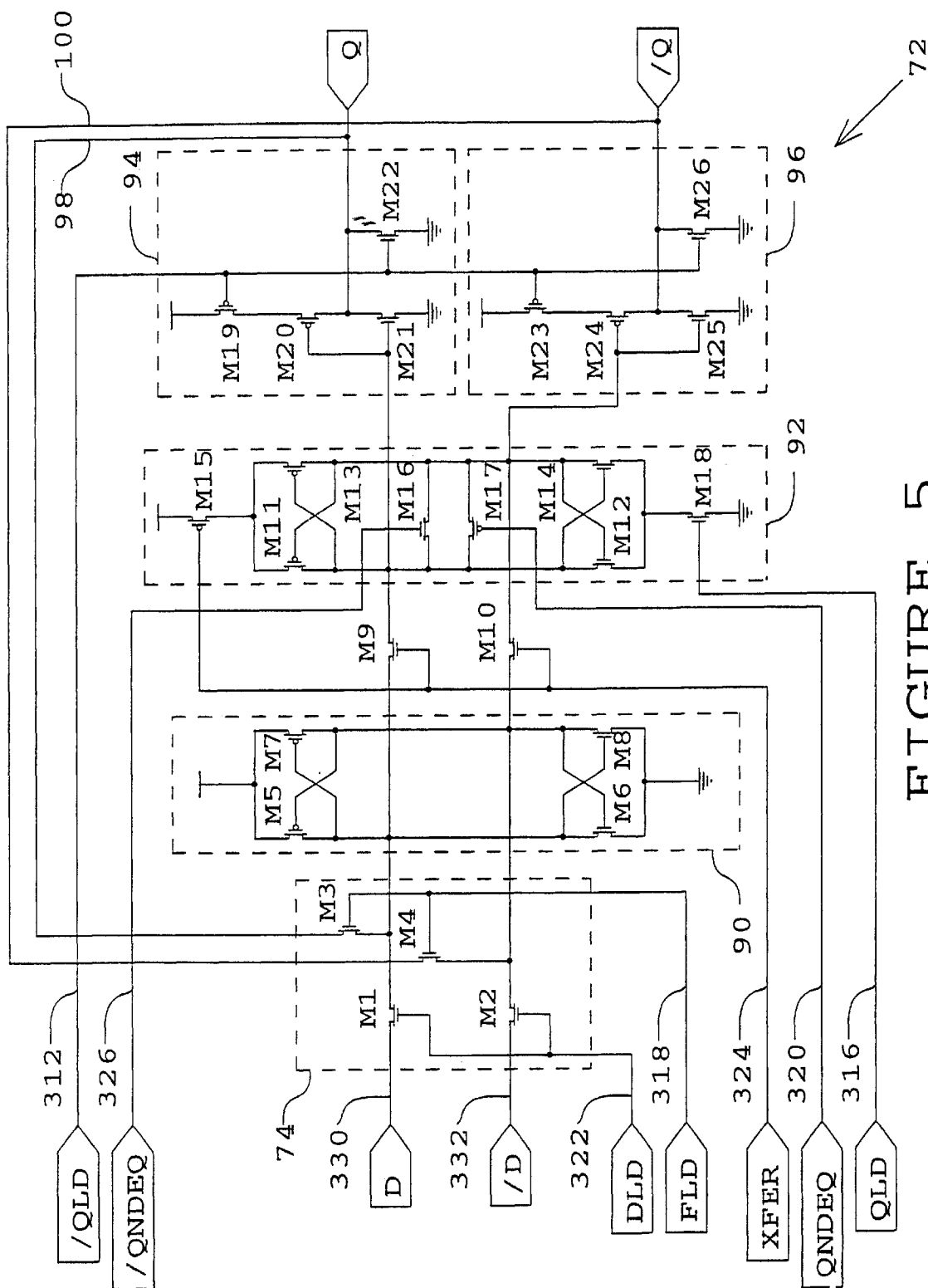


FIGURE 5

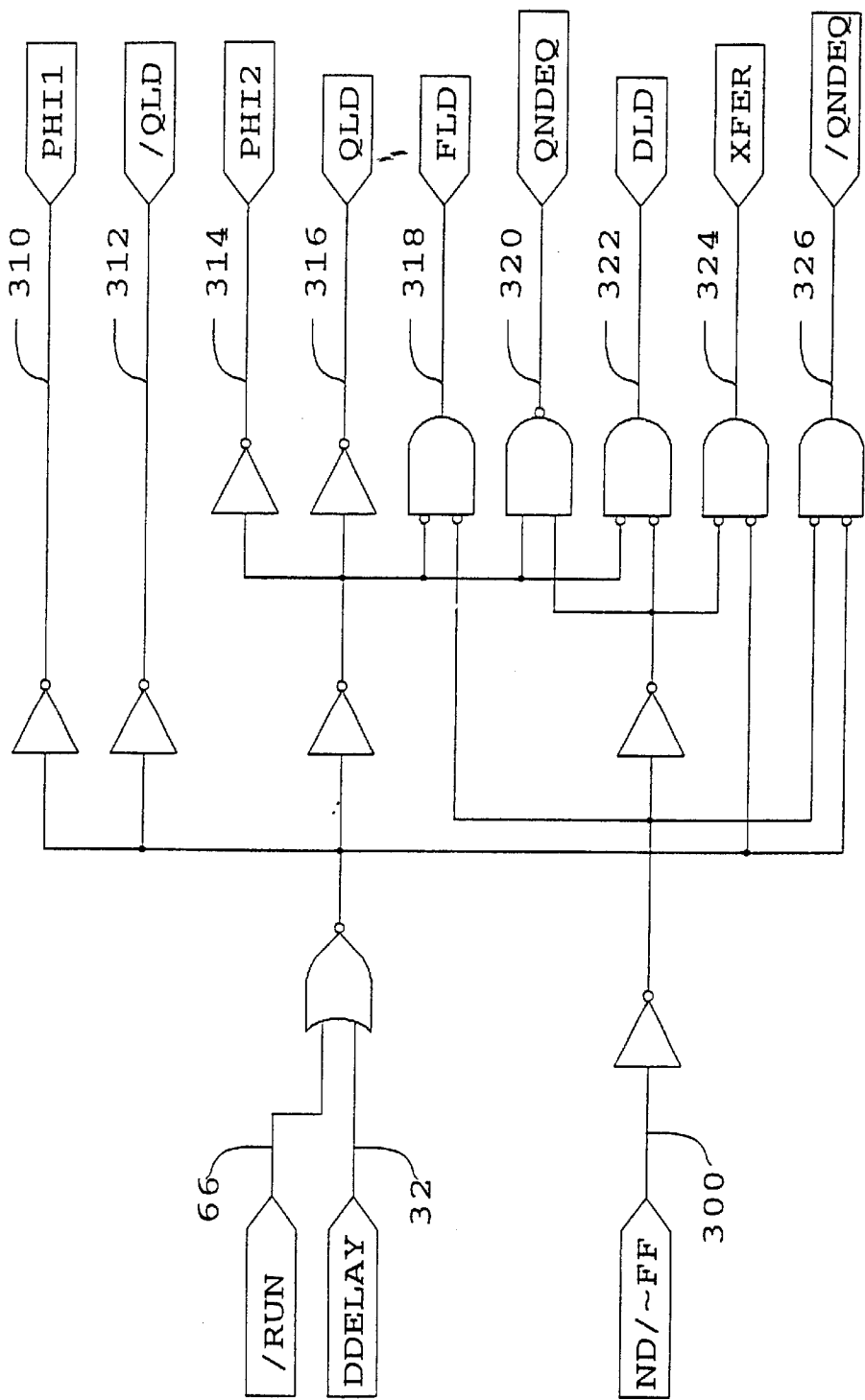
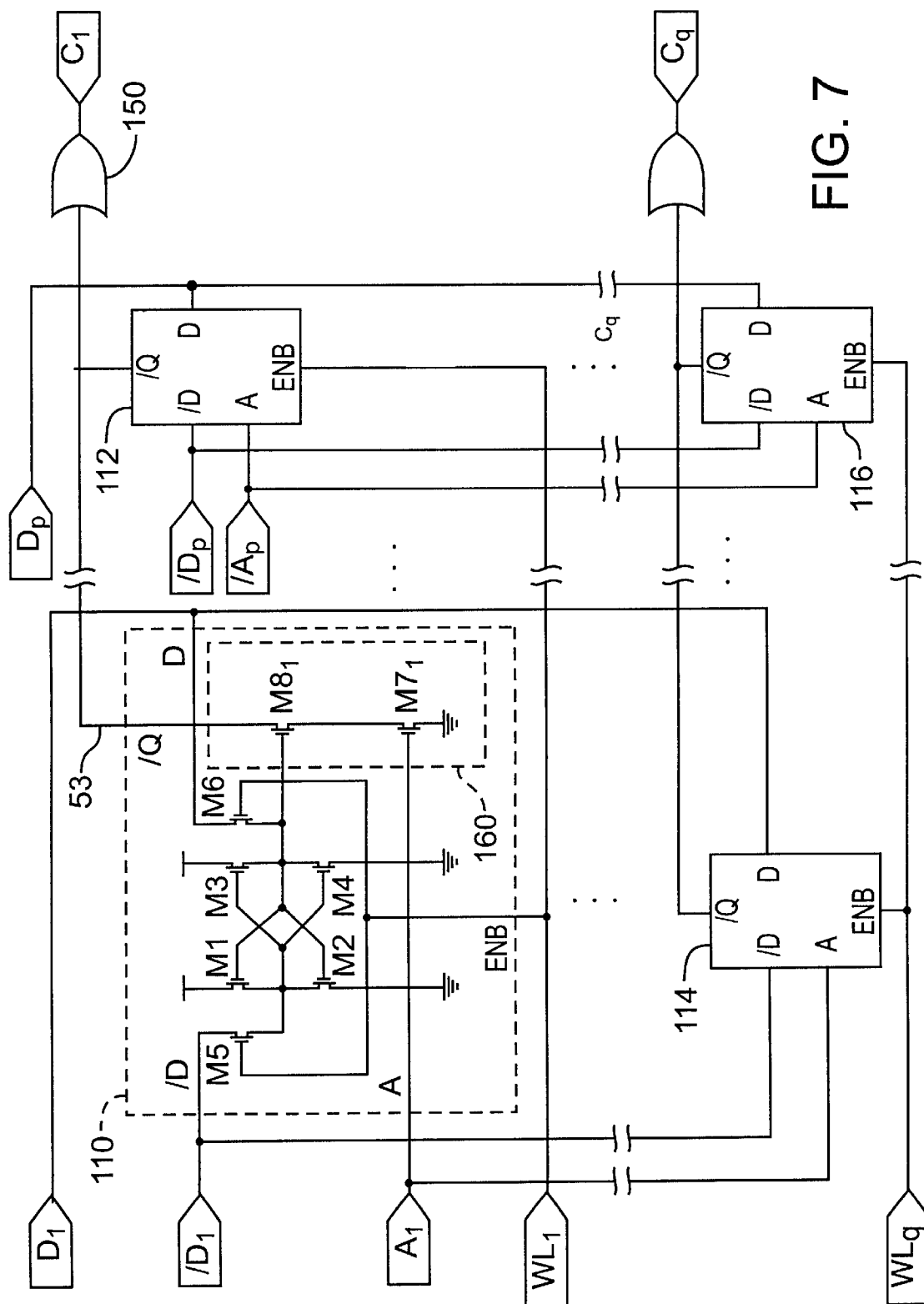


FIGURE 6



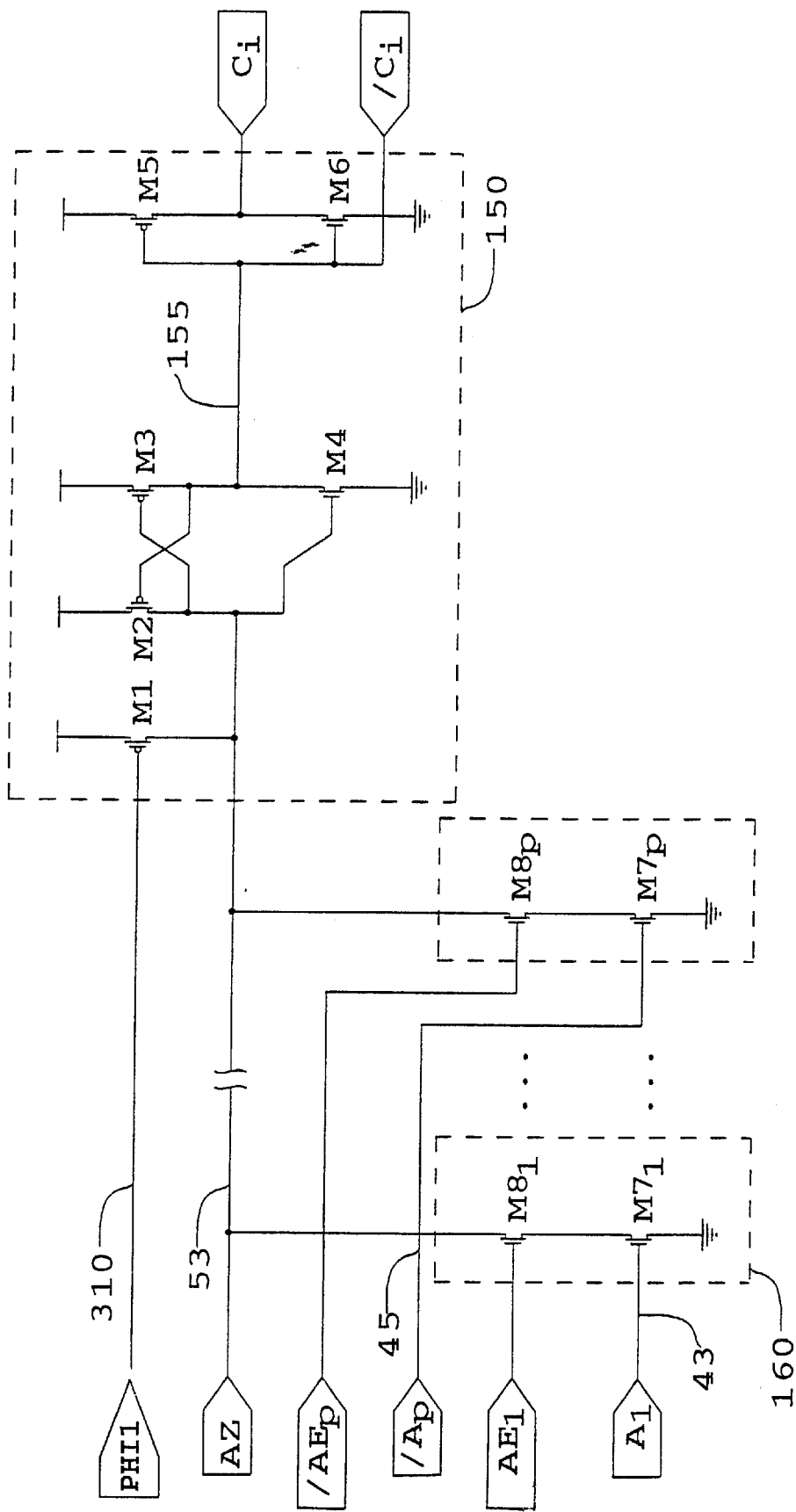


FIGURE 8

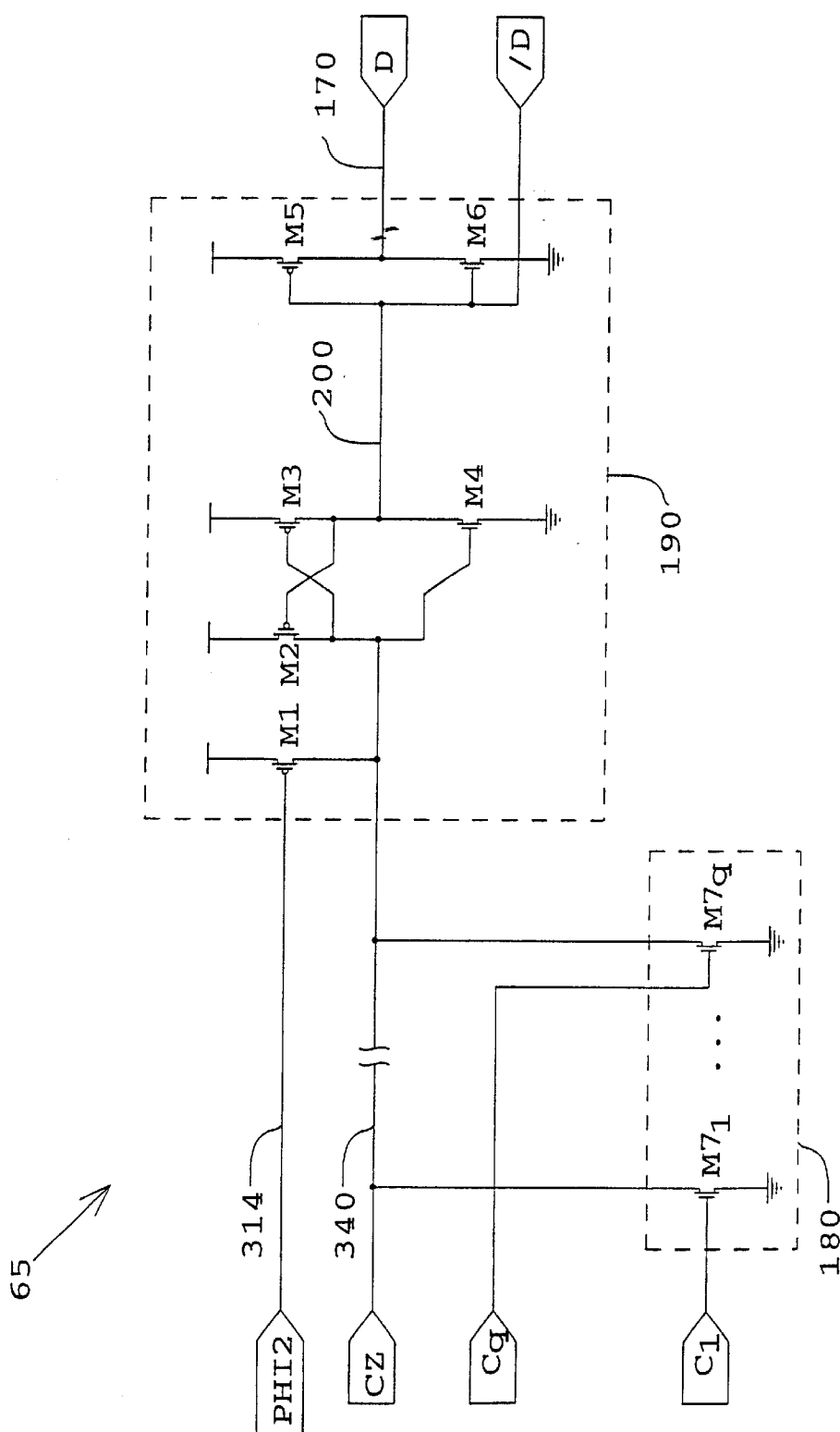


FIGURE 9

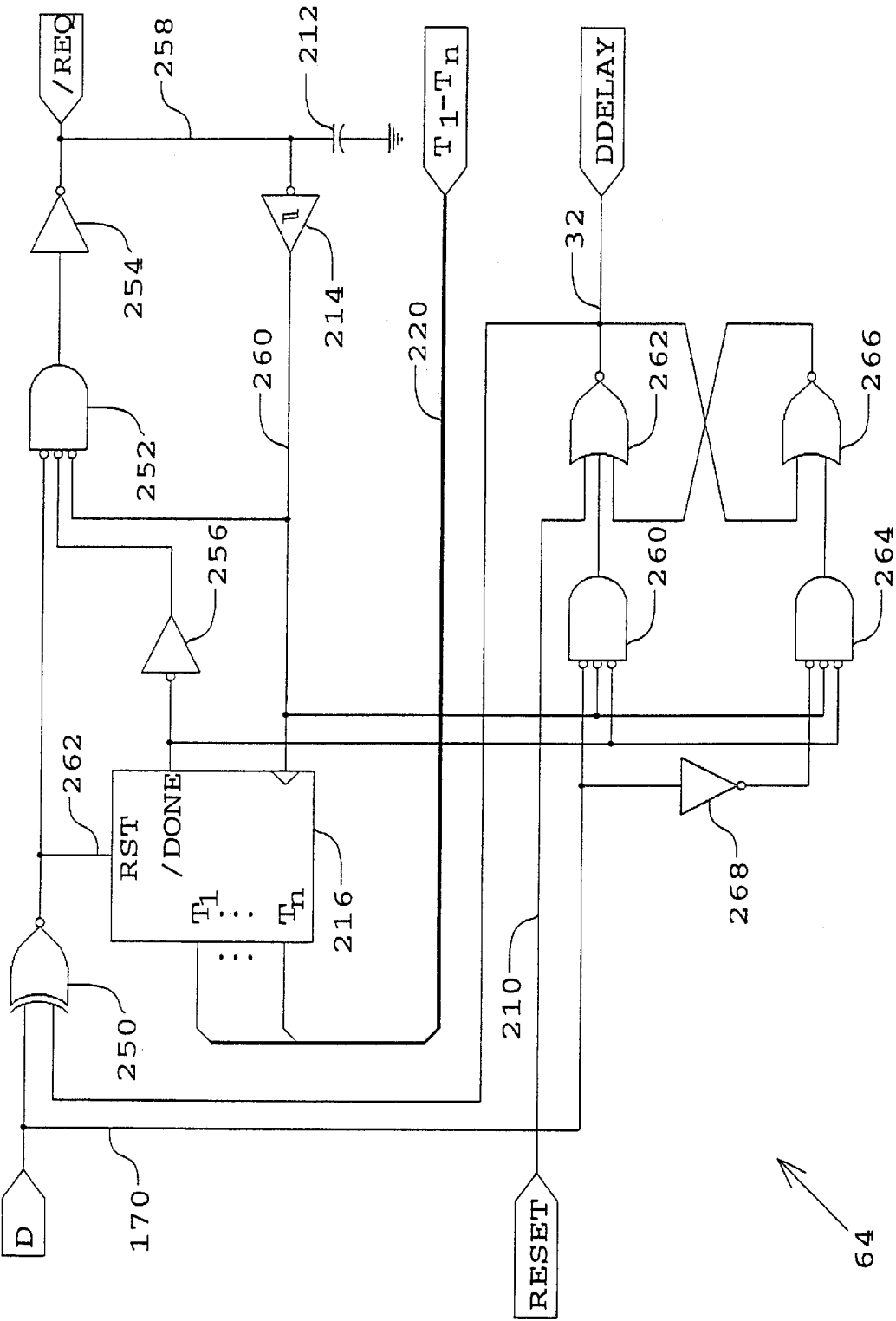


FIGURE 10

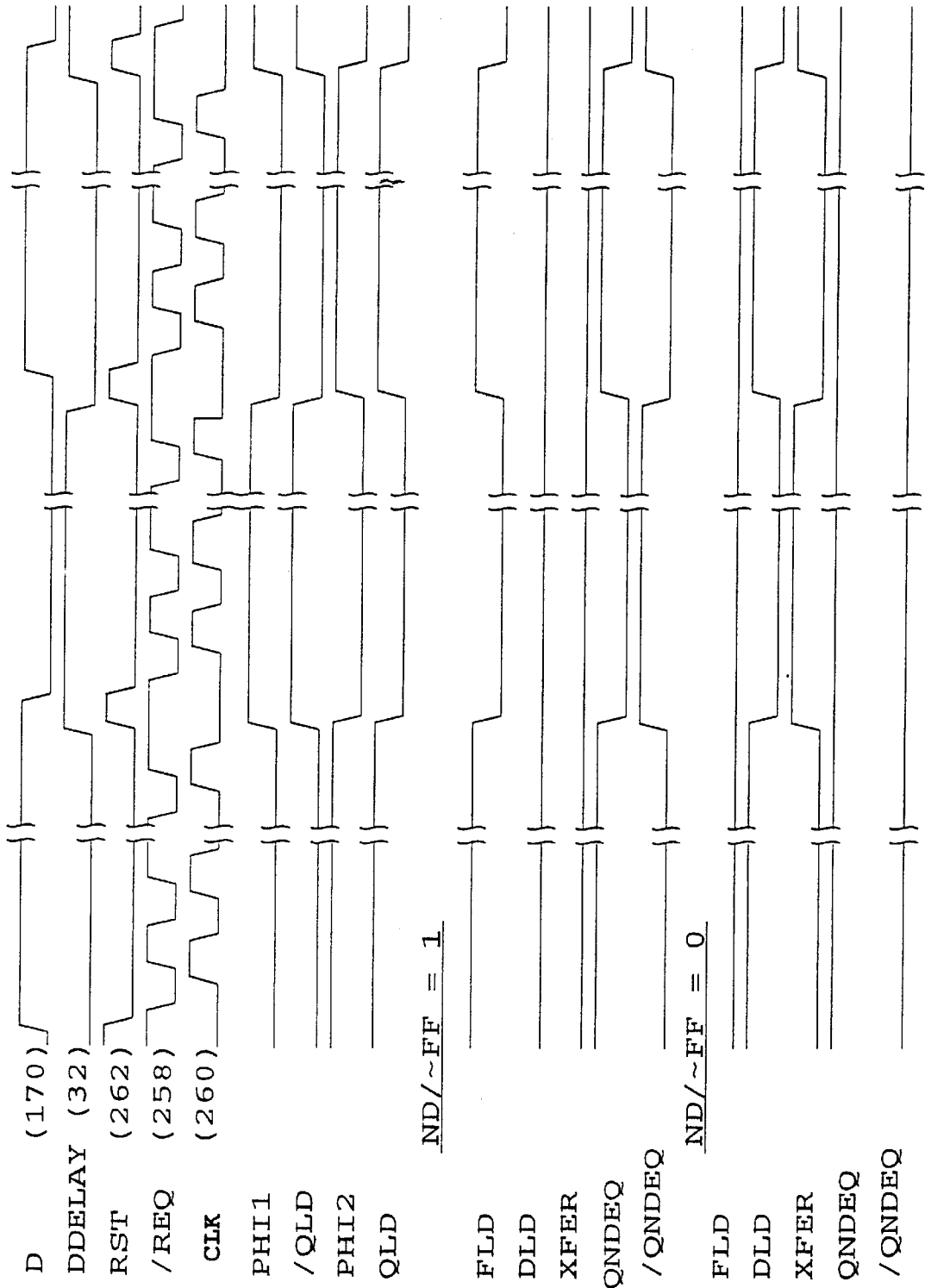


FIGURE 11

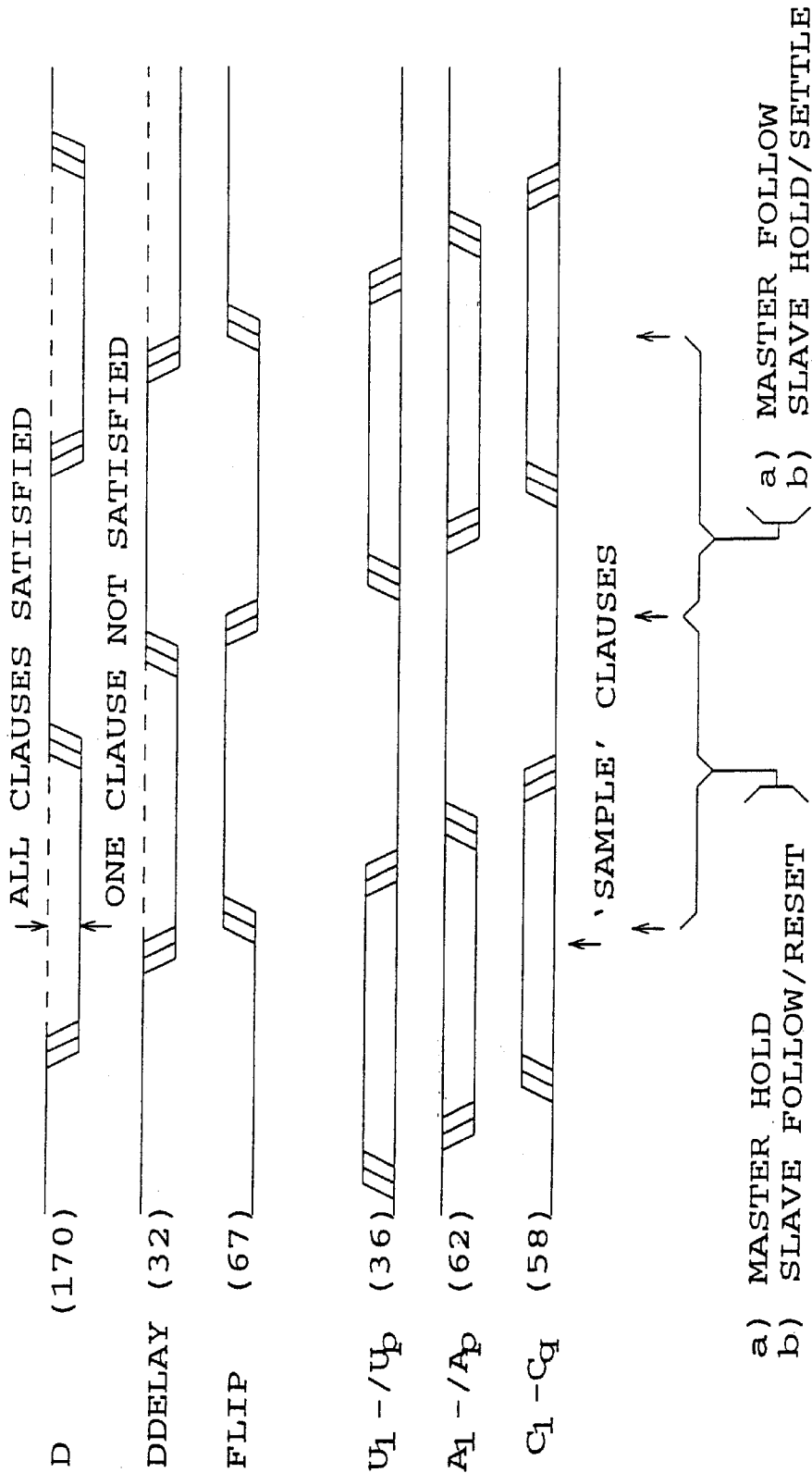


FIGURE 12

US 6,463,422 B1

1

PROBABILISTIC COMPUTING METHODS AND APPARATUS

This application is a continuation of application Ser. No. 08/296,654, filed Aug. 26, 1994, now U.S. Pat. No. 5,680, 518.

BACKGROUND OF THE INVENTION

For each combinatoric computing problem which focuses on finding an appropriate value assignment for the variables in the problem, a corresponding decision problem can be constructed which seeks a YES-NO answer as to whether a candidate assignment for the variables is actually a solution to the combinatoric problem. Nondeterministic Polynomial Time complete (NP-complete) decision problems are those for which no efficient solution method is known, in the sense that the number of steps in the solution method is a polynomial function of the size of the smallest representation of the problem (referred to as "polynomial-time"), but for which a candidate guess of the solution can be checked in polynomial time. It is known that all NP-complete problems are polynomial-time Karp-reduction equivalent, i.e., any instance of a particular NP-complete problem can be mapped in a polynomial time into an instance of some other NP-complete problem such that the solutions of this problem correspond to solutions of the original problem under the mapping.

Randomized computing methods incorporate probabilistic decision making techniques in addition to deterministic techniques and provide a solution to the problem which is correct with some minimum and selectable probability. Although it is believed that no deterministic, polynomial-time solution method exists for NP-Complete problems, it is known that the class PP of decision problems solvable in polynomial time by randomized computing methods includes the NP-complete problems. See "Structural Complexity I" by J. L. Balcazar, J. Diaz and J. Gabarro.

In the prior art, simulated annealing methods so-called Boltzmann machines have been referred to as randomized computing systems. See "Optimization by Simulated Annealing" by S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi; "A Thermodynamic Approach to the Traveling Salesman Problem: An Efficient Simulation" by V. Cerny, and "Boltzmann Machines: Constraint Satisfaction Networks that Learn" by G. E. Hinton, T. J. Sejnowski and D. H. Ackley. These methods differ from the present invention, however, in that they are more properly identified as heuristic methods for solving combinatorial computing problems. Although some versions of these methods do incorporate probabilistic decision techniques, the solutions they provide in polynomial-time are not guaranteed to be correct with some minimum selectable probability. The prior art does not appear to include any computing systems which efficiently realize randomized computing methods in the formal rigorous sense defined above.

SUMMARY OF THE INVENTION

The probabilistic computing system (PCS) of the present invention provides the computational functionality needed to efficiently realize randomized computing methods in otherwise standard, deterministic digital computing systems. The PCS may be incorporated in a standard computing platform such as a PC in various ways. For example, for applications requiring moderate performance, a VLSI probabilistic computation network chip, further described below, can be combined with standard dynamic RAMs and used as

2

a memory-mapped peripheral. In such a configuration, multiple PCS-DRAM modules can be interconnected for increased processing power. For higher performance applications, a peripheral device analogous to a mass memory peripheral (for example a hard disk) can be constructed by combining multiple PCS chips with dynamic RAM and interface logic. The disclosed invention allows solution of computing problems that heretofore could not be practically solved in small systems.

A probabilistic computing system according to the invention includes a memory for receiving and storing a digital representation of a predetermined combinatorial computing problem. The problem is expressed as a series of clauses in conjunctive normal form. In the preferred embodiment, the memory includes a plurality of rows of memory cells, each row arranged for storing a series of data bits corresponding to a respective clause of the computing problem. Each row stores a series of pairs of bits, each bit pair corresponding to a respective one of the variables and its logical complement.

A nondeterministic logic subsystem is provided for generating a set of random boolean values of the variables as a first proposed solution to the stored computing problem. Testing circuitry is coupled to the memory and to the nondeterministic logic means for testing whether the first set of random boolean values satisfies the stored computing problem. Finally, a feedback circuit couples the testing circuit and to the nondeterministic logic subsystem for controlling the ND subsystem so as to generate an alternative set of random boolean values of the series of variables as an alternative proposed solution to the computing problem whenever the testing circuit indicates that the computing problem is not satisfied. Thus the nondeterministic subsystem, the testing circuit and the feedback circuit together form a hardware loop for asynchronous operation. The system preferably is run asynchronously to maximize speed.

The nondeterministic subsystem includes a series of semiconductor nondeterministic logic elements. Each ND logic element includes a cross-coupled pair of inverting gates arranged for generating a respective one of the random boolean values of the series of variables, responsive to the variety of noise sources intrinsic to VLSI circuits.

The testing circuit includes, for each row of the memory means, a wide "programmable" NOR gate. The NOR gate provides a computed function signal (Cn) that indicates whether the proposed series of random boolean values satisfies the corresponding clause of the computing problem. Each bit of the data stored in the memory determines whether or not a corresponding one of the variables is included as an input to the NOR gate. In this way, the NOR gate is programmed to reflect the clause of the problem stored in that row of the memory.

Thus, the first wide NOR gate may be considered as part of a crosspoint switch array overlying the memory. The crosspoint switch array includes a series of crosspoint switch circuits, each crosspoint switch circuit being coupled to a respective bit of the corresponding row of the memory. Each crosspoint switch circuit also receives the corresponding variable signal from the corresponding ND logic element. In operation, each crosspoint switch circuit couples the corresponding probabilistic variable signal to a common circuit node—the NOR gate input—only if the said respective bit has a first predetermined logic state.

Accordingly, the programmable NOR gate in each row asserts the corresponding computed function signal responsive to such of the probabilistic variable signals as are

US 6,463,422 B1

3

coupled to the NOR input by the crosspoint switch circuits so as to indicate whether the first series of random probabilistic variable values satisfies the corresponding clause of the computing problem.

A second wide NOR gate is coupled to receive all of the computed function signals (C1–Cq) for determining and indicating whether the proposed solution satisfies the computing problem. This is true only when all of the computed function signals indicate that the corresponding clauses of the problem are satisfied. Accordingly, the second NOR gate has a number of inputs at least equal to the number of clauses in the computing problem. The output of the second NOR gate is used in the feedback circuit to trigger another set of values if the problem is not yet satisfied. A delay circuit in the feedback circuit provides for writing intermediate results to a DRAM.

The foregoing and other objects, features and advantages of the invention will become more readily apparent from the following detailed description of a preferred embodiment which proceeds with reference to the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a system level diagram of a probabilistic computing system according to the present invention.

FIG. 2 is a simplified block diagram of a computational data path of the system of FIG. 1.

FIG. 3 is a conceptual diagram of the C-RAM portion of the computational data path.

FIG. 4 is a simplified schematic diagram showing greater detail of a nondeterministic subsystem 60 of the computational data path.

FIG. 5 is a transistor level schematic diagram of an individual nondeterministic element of the nondeterministic subsystem.

FIG. 6 is a logic diagram of control logic for controlling the nondeterministic elements of FIGS. 4 and 5.

FIG. 7 is a partially-transistor level, partially-logic level diagram showing greater detail of the C-RAM of FIG. 3.

FIG. 8 is a transistor level schematic diagram showing detail of one row of crosspoint switch circuits and NOR gate 150 of the circuit of FIG. 7.

FIG. 9 is a transistor level schematic diagram of a q-bit wide NOR gate 65 of FIG. 4.

FIG. 10 is a schematic diagram of an asynchronous delay circuit generally corresponding to the delay element 64 of FIG. 4.

FIG. 11 is a timing diagram illustrating control signals generated by the control logic of FIG. 6 and FIG. 10 in operation.

FIG. 12 is a timing diagram illustrating overall operation of the probabilistic computing system according to the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

I. Introduction

The present invention is most useful for solving a combinatorial computing problem having a large number of variables. For example, if a computing problem has 1000 independent variables, a deterministic process for solving the problem may require either a very large computer in terms of memory space, or a very long time to carry out the computation. Probabilistic techniques essentially involve making a random “guess” at a solution, checking whether

4

the proposed solution satisfies the problem, and repeating this process until a solution is found or a time limit is encountered. We assume a problem expressed in conjunctive normal form, i.e. as a series of clauses, each clause made up of literals (variables or their complements) logically OR’ed together. A proposed solution (or “guess”) consists of defining the binary state or value of each one of the variables. Since these are chosen randomly, they will be called Probabilistic Variables (U1–Up, /U1–/Up) in the following description.

The heart of the apparatus described herein comprises a computational data path. In general, the computational data path includes a C-RAM and a Nondeterministic (“ND”) subsystem consisting of a set of ND elements, both described in detail later. One ND element is provided for each probabilistic variable. The ND elements generate binary random values (and their complements), i.e. the “guesses” and the C-RAM quickly checks whether a set of values (i.e. a proposed solution) satisfies the problem presented. Parallel, asynchronous operation of the computational data path provides high performance, while an interface is provided for convenient use in a conventional microcomputer system or workstation. This brief introduction oversimplifies both the apparatus and the methodology of the invention, but is provided to give the reader a starting point for understanding the following more detailed description.

II. Interface Structures

FIG. 1 is a system level diagram of a probabilistic computing system (PCS) according to the present invention. The system includes an external interface for interfacing with a host processor, a local DRAM array and other PCS devices, described as follows.

Three groups of interface signals for controlling and communicating with the PCS are provided. In the preferred embodiment, the PCS is implemented to be a memory-mapped peripheral device. Accordingly, the host processor interface is similar to a conventional, multi-bit wide DRAM interface. Referring to FIG. 1, parallel data for programming the PCS and for reading computation results are input and output on a data bus 12 comprising n+1 bits DB0–DBn. Addresses for accessing the internal RAM and the local external DRAM array are input on an address bus 14 comprising k+1 bits designated AB0–ABk. As shown in FIG. 1, buffered address signals from address bus 14 are input to control logic 15, and also are provided through a buffer 25 to the DRAM address bus 20 as bits DAB0–DABk.

Memory control signals 16 control the host processor interface. As with conventional DRAM, the upper and lower k+1 bits of a 2K+2-bit address are latched in the system by the row address strobe /RAS and the column address strobe /CAS. The transfer direction of an access operation is determined by the active low write enable /WE signal while the active-low output enable signal /OE gates output data to the data bus during read accesses.

To support a local DRAM array (not shown), the PCS also includes a local DRAM interface. Addresses for the local DRAM array are provided over DRAM address bus 20 as noted above. Data for the local DRAM array are provided over DRAM data bus DDB0–DDBn identified as bus 24. DRAM control signals include the DRAM row address strobe /DRAS, the DRAM column address strobe /DCAS, the active-low DRAM write enable /DWE and the active-low DRAM output enable /DOE collectively identified by reference number 28 in FIG. 1. These provide the control needed for reading and writing the local DRAM array under control of the control logic 15 as further described below.

The PCS optionally also includes a local expansion interface for networking multiple PCS systems to produce larger, higher performance systems. Each PCS system may be implemented as an individual integrated circuit, for example, or multiple systems may be integrated within a single device for increased density. Preferably, an array of PCS subsystems are arranged to interface with a host processor much like a mass-memory type peripheral system. The active low, open-drain /FLIP signal **32** (“FLIP FLAG”) serves as a control signal for synchronizing an array of PCS subsystems. As explained in detail below, the active low level of the /FLIP eventually induces a low-high transition on /FLIP which starts a new computation cycle. Thus, “wire OR”-ing together the /FLIP signal of all PCS chips in an array automatically synchronizes their operation.

Probabilistic variables **U1–Up** and their complements /**U1–Up** are bidirectional signals used to communicate data between the PCS chips in an array. These signals are communicated over bus **36**. The asynchronous nature of these signals means that various array architectures can be constructed depending on the application requirements of the array. The only requirement is that each chip in the array be programmed so that only one chip drives each signal pair corresponding to a variable. The same variables form part of the internal computational data path as will be seen below. Bus signals are summarized on the following Table:

TABLE 1

Bus Signal Definitions			
12	SYSTEM DATA BUS	DB0-DBn	n + 1
14	ADDRESS BUS	AB0-ABk	k + 1
20	DRAM ADDR BUS	DAB0-DABk	K + 1
24	DRAM DATA BUS	DDB0-DDBn	n + 1
58	COMPUTED FUNCTIONS	C1-Cq	q
36	PROBABILISTIC VARIABLES	U1-Up, /U1-/Up	2p
62	INVERTED PROBABILISTIC VARIABLES	A1-Ap, /A1-/Ap	2p

III. Computational Data Path

A. C-RAM

The computational data path, introduced above, is the portion of the PCS that asynchronously carries out the desired probabilistic computation. FIG. 2 is a simplified block diagram of the computational data path of the system of FIG. 1.

Referring to FIG. 2, the major components of the computational data path are a C-RAM **40** and a nondeterministic (hereafter “ND”) subsystem **60**. The ND subsystem **60** and related support logic are described in greater detail below with reference to FIG. 4.

Referring to FIG. 2, the ND subsystem **60** accepts as input on bus **58** a set of q bits or “Computed Functions” **C1–Cq** provided by the C-RAM **40** as explained below. Responsive to a /RUN signal **66** from control logic **15**, the ND subsystem **60** generates a new set of p binary random values and their complements, **U1–Up**, /**U1–Up**. These 2p values are inverted in a set of NOR gates **61**, and the resulting Inverted Probabilistic Variables **A1–Ap**, /**A1 – - /Ap** are provided over bus **62** to the C-RAM **40**, thus completing a computational data path loop.

FIG. 2 also shows how the control logic **15**, and address and data buses are coupled to the C-RAM **40**. RAM **42** receives control signals **38** from control logic **15**. These include address strobe /AS, write enable /WE and output enable /OE. RAM **42** also receives address data over address bus **30**. Address bus **30** includes upper address bits **UAB0–UABm** and lower address bits **LAB0–LABk** for a

total of k+m+2 address bits provided by control logic **15** as indicated in FIG. 1. RAM **42** also receives data bits **DDB0–DDBn** over data bus **12**. As indicated in FIG. 1, data bus **12** is coupled to the DRAM data bus **24** for data transfer with the local DRAM.

FIG. 3 shows the C-RAM **40** in greater detail. The C-RAM essentially comprises a 2p bits wide by q rows RAM **42**; a set of q 2p-input, “programmable NOR gates” referred to collectively as a crosspoint switch **50**; and a series of q NOR gates, for example NOR gate **44**, each corresponding to a respective row of the RAM, as described in detail later. By loading the RAM with the proper data, the crosspoint switch **50** is “programmed” to compute the logical NOR function of a desired subset of the inverted input variables **A1–Ap**, /**A1 – - Ap**. (Note: the C-RAM actually computes the NOR function of the appropriate variables. The results of these computations are then NOR’ed together by NOR gate **65** to produce the D signal **170**. This is logically equivalent to a CNF formula for D in which the appropriate variables are ORed together to produce the functions /C1, /C2, . . . /Cq and then ANDing these signals together to produce D.) Each clause is “programmed” in a corresponding row of RAM **42**. Thus, each NOR gate corresponds to one clause of the computing problem presented. Each column in the RAM corresponds to one of the inverted probabilistic variables or its complement. Operation of the C-RAM is described in greater detail later in connection with a description of the preferred circuitry.

The following example shows how a problem expressed in conjunctive normal form is stored in the C-RAM. Conjunctive normal form is a series of clauses logically ANDed together, each of the clauses consisting of a series of variables, or literals, logically ORed together. A simple example is:

$$(U2 \text{ OR } U5 \text{ OR } /U6) \text{ AND } (U1 \text{ OR } /U3) \text{ AND } (/U2 \text{ OR } /U4 \text{ OR } U6).$$

In this example, there are three clauses. Accordingly, this problem is stored in three rows of the C-RAM. In general, the RAM data format is: **A1 /A1 A2 /A2 A3 /A3 . . . Ap /Ap** where **A(x)** is the inverted probabilistic variable. So, for example, the first row in RAM, representing the first clause in the example above, contains the following data:

RAM contents at row 1: 001000001001000000 . . .

Assuming the foregoing RAM contents, crosspoint switch **50** will connect inputs **A2** and **A5** and /**A6** together (wired-OR) as input to the first row NOR gate. The output of the NOR gate is called a Computed Function **C(1)** (true=0). As shown in FIG. 3, all of the row NOR gates are disposed in parallel. All of the values are generated in parallel. All of the NOR gate outputs, computed functions **C1–Cq**, are output in parallel on bus **58** to the ND subsystem. The computed function is low for each clause which is satisfied.

B. Nondeterministic Subsystem

FIG. 4 shows the ND subsystem **60** in greater detail. The computed functions **C1–Cq** are OR-ed together in NOR gate **65** to provide a signal **D** at node **170**. Since each computed function **C(n)** is low if true, then all of them will be low only if all clauses of the problem stored in the C-RAM are satisfied. Accordingly, **D=1** indicates a satisfactory solution to the whole problem. **D** is input to a delay circuit **64**, described below, which in turn provides DRAM timing signals on bus **66** (**T1–Tn**) and output signal **DDELAY** which, through a buffer **68**, provides the synchronization signal /FLIP **32**.

The ND subsystem further includes a series of p individual ND logic elements **70**, for example ND logic element **72**. Each individual ND logic element driven by the variety of noise sources intrinsic to VLSI circuits, generates one of the probabilistic variables and its complement. For example, ND element **72** provides **U1**, **/U1** as its outputs. These signals **U1**, **/U1**—**Up**, **/Up** are output in parallel on bus **36**. The **DDELAY** signal fans out through a series of inverters **80** to form a series of control signals, e.g. at node **67**, each control signal being coupled to a corresponding one of the individual ND elements. Control of the ND elements is described in detail later with reference to the associated control logic and timing diagrams.

FIG. **5** is a transistor level schematic diagram of a representative one of the individual ND logic elements **72**. Each ND logic element comprises an input multiplexer section **74**, followed by a master stage **90**, followed by a slave/ND stage **92**, followed by output NOR gates **94,96**. This circuit operates somewhat similar to a D-type flip-flop. It may be configured to follow data and then hold, responsive to a clock-type input, or to hold data and then follow. The “follow mode” is described first.

1. Predetermined (Deterministic) Mode of Operation

The circuitry shown and described can be configured to generate random values for all of the variables **U1–Up**. In the preferred embodiment, however, provision is made for a subset of the variables to be predetermined. Accordingly, the circuit of FIG. **5** provides master-slave operation to allow loading predetermined values for selected variables. Where such data is arriving from outside the PCS, the ND elements can be partitioned since only part of the data is nondeterministic. Those ND logic elements corresponding to predetermined values function as latches. To do so, master stage **90** followed by SLAVE/ND stage **92** provide a latch function, responsive to a transfer control signal **XFER** controlling the gates of transfer transistors **M9,M10**, control of gating transistors **M15** through **M18**, as further described below. This is called the deterministic or flip-flop mode of operation. (All of the variables are included in **U1** through **Up** and denominated “probabilistic variables” for simplicity.) In the flip-flop mode, the ND element generates new data by simply latching the input data **D** and **/D** received externally. Referring to FIG. **1**, bus **31** is used to load data from the external DRAM into the ND subsystem **60**. Referring again to FIG. **5**, this data is input via the **D** (and **/D**) inputs.

2. Nondeterministic Mode of Operation

The nondeterministic mode of operation of the circuit of FIG. **5** is described next. In this mode, the ND element holds the previous random decision. Referring to the slave/ND stage **92**, transistors **M11** and **M12** form a first inverter and transistors **M13–M14** form a second inverter, arranged as a cross-coupled pair. Transistor **M18** controls power to the cross-coupled pair. Transistors **M16** and **M17** are charge equalization transistors, together forming a transmission gate—a functional equivalent of a wire when both are turned on. In nondeterministic operation, transistors **M16** and **M17** are turned on while the power (**M18**) is turned off, so as to equalize charge on the gates between transistors **M11** and **M13**. Thereafter, once transistors **M15** and **M18** are turned on and transistors **M16–M17** are turned off, thermal noise will cause the cross-coupled pair of inverters to assume one of two stable states with unknown probability. The final state of the cross-coupled pair is then a probabilistically selected random value.

Transistors **M19, M20, M21** and **M22** together form a first NOR gate **94** which provides the **Q** output signal. Transistors

M23, M24, M25 and **M26** form a second NOR gate **96** that provides the **/Q** output signal. These NOR gates serve as inverters to force an inversion when the circuit is in an undetermined state. This has the effect of an ensuring valid logic levels as well as providing inversions needed for other parts of the circuit. Inverters could be used in this application but NOR gates are believed to provide better reliability. Each **Q** output provides a corresponding one of the probabilistic variables **Un** and the **/Q** output provides the complement **/Un**.

In a practical implementation in an integrated circuit, individual ND elements could be grouped, for example in groups of 8 or 16 such elements, and the control logic of FIG. **6** would be replicated for each group of ND elements. Each group would operate in either the deterministic (flip-flop) or nondeterministic mode for a given problem. The mode is selected by a ND/FF control signal (input at node **300** FIG. **6**) which is provided by the control circuitry. So, for example, if 10 deterministic bits are required, the control circuitry would switch two groups of 8 ND elements to the deterministic mode of operation (FF). The “extra” 6 bits would simply not be connected in the C-RAM. Those bits would be excluded from all the clauses in the problem.

3. ND Element Timing and Control Logic

Transistors **M1** and **M2** provide a multiplexer function for selecting between deterministic inputs **D** and **/D** or a nondeterministic value which is fed back from the **Q** output via feedback path **98** and from the **/Q** output via feedback path **100**, respectively, for latching the last nondeterministic decision. A control signal **DLD** (“deterministic load”) indicates flip-flop mode of operation and gates external deterministic data **D** and **/D** into the circuit, while a mutually exclusive control signal **FLD** (“feedback load”) indicates feedback operation and controls transistors **M3–M4** to feedback the **Q** and **/Q** signals.

FIG. **6** illustrates logic circuitry to provide control signals for operating the circuitry of FIG. **5** in the manner described. Operation of the logic of FIG. **6** will become apparent when the logic diagram is taken together with the signal definitions shown in the following Table 2 and timing diagram of FIG. **11**.

TABLE 2

Control Signal Definitions	
D (/D)	deterministic input bit
DLD	deterministic load signal (flip-flop mode)
FLD	feedback load signal (ND mode)
XFER	master-slave transfer control signal
QNDEQ (/QDNEQ)	charge equalize control signal slave/ND stage
QLD (/QLD)	SLAVE/ND stage charge control signal
PHI1 (FIG.8)	precharge control for p-bit wide NOR gates 150
PHI2 (FIG.9)	precharge control for q-bit wide NOR gate 65

IV. Detailed Circuit Descriptions

FIG. **7** is a diagram showing greater detail of the C-RAM of FIG. **3**. In this diagram, each bit of the C-RAM circuitry is represented by a box, for example bit circuit **112** is the C-RAM bit circuit at row **1**, column **p**, while bit circuit **114** is the C-RAM bit circuit at row **q**, column **1**. Thus, the C-RAM comprises a rectangular array of individual bit circuits, the array having rows **1** through **q** (corresponding to **q** clauses) and columns **1** through **p** (corresponding to **p** variables). The row **1**, column **1** bit circuit **110** is illustrated at the transistor level. Transistors **M1, M2, M3, M4, M5** and **M6** form a static memory cell. Writing the cell is controlled by an enable signal on a corresponding word line **WL**. Data

US 6,463,422 B1

9

for storage in the C-RAM memory cells are provided at data inputs D1 through Dp (and the respective complement inputs). The data, representing the problem of interest expressed as described above, may originate on the system data bus 12 (FIG. 1) or DRAM data bus 24 (FIG. 1). Operation of a memory cell of the type shown is known. Various other known memory cells can be used, depending on the applicable fabrication process, and trading off speed, power, density, etc. All such variations are merely engineering design choices and shall be considered equivalents within the scope of the present invention.

Each C-RAM bit circuit includes a crosspoint switch circuit. In bit circuit 110, transistors M71 and M81 form a crosspoint switch circuit 160. The data bit stored in the cell (reflected at the gate of M81) controls transistor M81 so as to couple (or not couple, depending on the state of the cell) the corresponding inverted probabilistic variable signal A1, via transistor M71, to an output node 53. Similarly, each of the other bit circuits in row 1 selectively couples a respective one of the remaining variable signals, A2 . . . /Ap to the row output node 53. All of the bit circuit outputs of the row are wired-ORed together at node 53, which in turn is input to an OR-gate 150 to provide the corresponding computed function output C1. Gate 150 actually functions as a NOR gate because each bit circuit output /Q is inverted and the NOR gate input is normally pulled high, as explained below. In the same manner, each of the other rows 2 . . . q of the C-RAM array selectively couples the inverted variable signals A1 . . . /Ap to a corresponding row output node. The row output node is input to a corresponding NOR gate to form the corresponding computed function C1 . . . Cq. In operation, each computed function indicates whether or not the clause computed in the corresponding row of the C-RAM is satisfied.

Referring now to FIG. 8, one row of crosspoint circuits, for example crosspoint circuit 160 (comprising transistors M71 and M81), is shown coupled to NOR gate 150. Memory enable signals AE1 through /AEp are coupled to the corresponding crosspoint switch circuits, respectively. These signals reflect the bits stored in the C-RAM. The inverted probabilistic variables A1 through /Ap are coupled to the crosspoint switch circuits, respectively, as noted above. The outputs of all of the crosspoint switch circuits in the row are coupled together at node 53 (also labeled AZ), and input to NOR gate 150. Identical circuitry is provided for each row of the C-RAM.

FIG. 8 also shows the NOR gate 150 in greater detail. Transistor M1 serves to precharge the input line 53 for resetting the gate, under control of the precharge control signal PHI1 at node 310. Transistors M2, M3 and M4 essentially form a latch, while transistors M5 and M6 provide an inverting output buffer circuit. In operation, node 53 is normally high, and is pulled low if any of the individual crosspoint circuits is on or true. Each crosspoint circuit essentially forms an NAND gate with open drain, its output being true (low) only if the corresponding enable signal (AEx) is true (i.e. the corresponding C-RAM memory bit is a one) and the corresponding variable (Ax) is true. Thus, a particular crosspoint switch will be on where the corresponding variable is included in the corresponding clause stored in the C-RAM, and that variable is true in the proposed solution. If any one or more of the crosspoint switches in a row is ON, node 53 is pulled low, and therefore the corresponding computed function C goes low, indicating the corresponding clause is satisfied, since each clause in the CNF formula is an OR function of the variables.

FIG. 9 is a transistor level schematic diagram of the q-bit wide NOR gate 65 of FIG. 4. It may be recalled that NOR

10

gate 65 receives as inputs all of the computed functions C1 . . . Cq (bus 58). Referring to FIG. 9, the computed functions C1 . . . Cq are applied to the gates of transistors M71 through M7q, respectively, in an input stage 180. The wired-OR function of all of these input signals appears at node 340 (CZ). CZ is input to OR gate 190. Gate 190 is essentially the same in form and function as gate 150 described above, so the details will not be repeated. The output of gate 190 provides the D signal at node 170 as well as the complement /D. This is the same D signal identified in FIG. 4 at the input to delay circuit 64. D is asserted (hi) if all clauses are satisfied. If any clause is not satisfied (C is high), then CZ and ergo D go low.

FIG. 10 is a schematic diagram of an asynchronous delay circuit generally corresponding to the delay circuit 64 of FIG. 4. The circuit of FIG. 10, after a predetermined delay period described below, asserts the signal DDELAY to commence a new cycle. Specifically, the reason for this delay circuit is to allow for storing interim data into the external DRAM after each cycle. At least a non-zero delay is necessary for the PCS circuit to function correctly. As a practical matter, the delay for writing to DRAM far exceeds the minimum required delay, which is an inverse monotonic function of the analog gain of the computational data path, so it is not an issue. For some applications where interim results are not required, the delay circuit can be omitted. In that case, parasitic capacitance will provide the minimum delay, while allowing the PCS circuit to run asynchronously at maximum speed.

Logic gates 252, 254 and 214 form an inverting loop. The Schmitt trigger 214 converts a capacitor 212 voltage to proper logic levels so that these components together form a free-running oscillator loop, as illustrated by the /REQ and CLK signals in the timing diagram of FIG. 11. The size of capacitor 212 is selected so as to provide an appropriate delay period for writing to the DRAM. The CLK signal appears at node 260, the Schmitt trigger output. Each cycle of this oscillator clocks counter 216, thereby incrementing its outputs T1-Tn at bus 220. Signals T1 - - - Tn generally indicate DRAM timing and control signals, the particulars of which will depend on the specific implementation. For example, if the PCS provides for 1028 variables, and the DRAM is 8 bits wide, then counter 216 can be arranged to count to 256 before overflow. In this case, T1-Tn can be arranged to provide strobes and addresses for storing the variables in 128 DRAM locations. When counter 216 overflows, the /DONE output goes low at the inputs to gate 256 and 264. Gate 256 drives gate 252 to stop the oscillator. The present state of the nondeterministic variables (the proposed solution) has been stored in the DRAM.

Logic gates 260, 262, 264, 266 and 268 together form a C-gate circuit. The C-gate receives the D input 170, the /DONE signal from counter 216 and the CLK signal 260. The C-gate output at node 32, DDELAY, controls the start of a new cycle. It reflects whatever state appeared at D changes state on the next clock cycle after the counter overflows and D changes state. A RESET control signal at node 210 forces start of a new cycle. It is provided by the control logic, e.g. to begin a computation. DDELAY also is fed back to gate 250 to counter 216 reset input 262 and oscillator gate 252.

VI. Summary of Operation

Operation of the PCS system is summarized with reference to the timing diagrams of FIGS. 11 and 12. Referring to FIG. 12, a new set of probabilistic values U1-Up are generated in the ND subsystem, some of which may be predetermined. FIG. 11 shows the control signal timing for both the ND elements operating in nondeterministic mode

US 6,463,422 B1

11

(ND/~FF=1) and those operating in the deterministic mode (ND/~FF=0). Inverted variables A1~Ap follow by a few gate delays and are applied to the C-RAM. There the computed functions C1~Cq are formed, and these determine a new state of D. D is low, indicating that at least one clause is not satisfied. After delay, DDELAY goes low and /FLIP goes high, to synchronize and begin a new cycle. Note that the system runs asynchronously. For example, the inverted variables pass through the cross-point switch with only a few gate delays, and each computed function C will change state as soon as the corresponding clause is first satisfied (even if some bits lag behind). Similarly, D goes low as soon as one computed function indicates that the corresponding clause is not satisfied; it does not wait for other clauses to be determined. These features have the advantage of very high speed of operation. As noted, significant delay is inserted only as necessary for saving interim results.

The time complexity of the computation carried out by the system is summarized as follows. Due to processing imperfections, it should be expected that each of the probabilistic variables A1, A1 . . . Ap is biased and correlated. Ignoring the correlation effects for the moment, let M represent the set of 2p possible assignments for the probabilistic variables. The probability that the system finds a solution on a single attempt is:

$$Q = \Sigma P[\text{occurrence of } m]$$

m in M

To accommodate the effect of biases, it is reasonable to assume that an upper bound Q' exists on the probability that the system does not find a solution on a single attempt:

$$1 > Q' = q(2p-m) \geq 1 - Q$$

It follows that a lower bound on the probability that the system with p variables finds a solution in k attempts to a problem with m solutions is:

$$P(p, m, k) = 1 - [q(2p-m)]^k$$

Therefore, the number of attempts K needed to insure that a solution is found with probability P is:

$$K > \frac{\log(1-P)}{\log(2p-m) + \log q} = \frac{\log[1/(1-P)]}{\log 1/q - \log(2p-m)}$$

To study this result further, it is reasonable to assume that 1/q scales exponentially with p:

$$1/q = 2^{p/r}$$

so that

$$K > \frac{\log[1/(1-P)]}{\log r + p - \log(2p-m)}$$

It follows immediately that a solution is found in a constant number of tries determined solely by P and m if m scales as

$$m > 2^{p-1}$$

since

$$\log(2^p - m), \log 2^{p-2^{p-1}} = p-1$$

12

and therefore

$$\frac{\log[1/(1-P)]}{\log r + 1} = \frac{\log[1/(1-P)]}{\log r + P - \log(2^p - m)}$$

As mentioned earlier, in addition to biases, it should also be expected that successive assignments for each of probabilistic variables A1, A2 . . . Ap will be correlated to some degree. It is the nature of VLSI physics, however, that the correlation between the k-th and the j+k-th attempt decreases rapidly with j and is virtually nil for some relatively small J. As a result, every J-th attempt can be considered to be uncorrelated and the number of attempts needed to insure that a solution is found with at least probability P is JK (which is still constant),

V. Alternative Embodiments

The circuitry described above for probabilistic computing could be implemented together with a microprocessor in a single integrated circuit. The current state of the art in terms of density and memory size might require sacrificing some of the parallelism of the C-RAM. In such an alternative embodiment, individual clauses or even parts of clauses may be evaluated serially due to memory and gate density limitations. Another alternative would be to implement the probabilistic circuitry in a stand-alone special purpose processor. Again, depending on the process, technology, etc., such an implementation may or may not include adequate RAM on board for parallel processing multiple clauses. Various integration schemes, trading off performance, packaging density, power etc. may be used, all of which involve design choices that fall within the scope of the present invention and thus are to be considered equivalents of the embodiments discussed herein.

A further alternative embodiment is to implement the functionality described above using commercially available off-the-shelf programmable logic devices such as for example, the Intel flexlogic family of PLDs that include bistable set-reset storage elements in which both set and reset can be programmed to be active simultaneously and for which neither set or reset is dominant, (i.e., activating both set and reset puts the storage element in an unstable state rather than the set or reset state). The necessary circuitry could be implemented in such PLDs combined with adequate external memory to store the data that would otherwise appear in the C-RAM. PLD implementations should be considered equivalent to the preferred embodiment disclosed above in greater detail.

Having illustrated and described the principles of my invention in a preferred embodiment thereof, it should be readily apparent to those skilled in the art that the invention can be modified in arrangement and detail without departing from such principles. I claim all modifications coming within the spirit and scope of the accompanying claims.

I claim:

1. A nondeterministic logic circuit for generating random boolean values of one or more variables as a proposed solution to a computing problem expressed in conjunctive normal form as one more clauses in said one or more variables, the logic circuit comprising:

one nondeterministic logic element for generating a respective random boolean value for each one of the said one or more variables; and

each nondeterministic logic element comprising:

a cross-coupled pair of transistor inverter circuits;

means for controlling power to the cross-coupled pair of transistor inverter circuits; and

means for equalizing charge on the gates of the transistor inverter circuits while power is removed from

US 6,463,422 B1

13

the cross-coupled pair, thereby driving the cross-coupled pair to an unstable equilibrium, whereby intrinsic circuit noise will cause the cross-coupled pair to randomly assume one of two stable states when power is restored to the cross-coupled pair, the stable state assumed by the cross-coupled pair providing a probabilistically selected random boolean value

and further comprising common synchronization means coupled to all of the nondeterministic logic elements for synchronizing operation of the nondeterministic logic elements.

2. A hardware probabilistic computing system comprising:

memory means for receiving and storing a digital representation of a predetermined combinatorial computing problem expressed as a series of clauses in conjunctive normal form, the memory means including a plurality of rows of memory cells, each row arranged for storing a series of data bits corresponding to a respective clause of the computing problem;

nondeterministic logic means for generating a first set of random boolean values of a series of variables as a first proposed solution to the stored computing problem, the nondeterministic logic means including a series of semiconductor nondeterministic logic elements, each nondeterministic logic element arranged for generating a respective one of the random boolean values, and the series of semiconductor nondeterministic logic elements being coupled together so as to generate the random boolean values for all of the variables substantially concurrently in response to a single flip control signal;

testing means coupled to the memory means and to the nondeterministic logic means, the clause testing means including, for each row of the memory means, first logic means coupled to the corresponding row of the memory means and coupled to the nondeterministic logic elements for providing a respective computed function signal that indicates whether the series of random boolean values satisfies the corresponding clause of the computing problem;

second logic means coupled to receive all of the computed function signals for determining and indicating that the first proposed solution does not satisfy the computing problem as soon as any one of the computed function signals indicates, by its logic state, that the corresponding clause of the problem is not satisfied; and

feedback means coupled to the second logic means for asserting the flip signal to generate an alternative set of random boolean values as an alternative proposed solution to the computing problem responsive to the second logic means indicating the first proposed solution does not satisfy the computing problem, so that the nondeterministic logic means, the clause testing means, the second logic means and the feedback means together form a hardware loop operable asynchronously.

3. A probabilistic computing system according to claim 2 wherein:

the first logic means comprises a series of crosspoint switch circuits, each crosspoint switch circuit being coupled to receive a respective bit of the corresponding row of the memory means;

each crosspoint switch circuit further being coupled to receive the corresponding variable signal;

14

and each crosspoint switch circuit including means for coupling the corresponding probabilistic variable signal to a common circuit node only if the said respective bit has a first predetermined logic state; and

the first logic means further comprises a logic gate coupled to the common circuit node for asserting the computed function signal responsive to such of the probabilistic variable signals as are coupled to the common circuit node by the crosspoint switch circuits so as to indicate whether the first series of random probabilistic variable values satisfies the corresponding clause of the computing problem.

4. A probabilistic computing system according to claim 4 wherein the second logic means comprises a NOR gate having a number of inputs at least equal to the number of clauses in the computing problem.

5. A probabilistic computing system according to claim 2 wherein:

the testing means includes an array of logic gates and a crosspoint switch array controllably interconnecting the nondeterministic logic means and the logic gates, the crosspoint switch array also being coupled to the memory means so that the crosspoint switch array couples selected ones of the said variables to the logic gates responsive to the representation of the computing problem stored in the memory means, whereby each logic gate computes a logical function of a subset of the variables, the subset being defined by data stored in the memory.

6. A probabilistic computing system according to claim 2 further comprising a DRAM type of interface means for storing intermediate results in a second memory.

7. A probabilistic computing system according to claim 2 further comprising a DRAM type of interface means for interfacing the probabilistic computing system to a host processor as a memory-mapped peripheral device.

8. A hardware probabilistic computing system comprising:

memory means for receiving and storing a digital representation of a predetermined combinatorial computing problem expressed as a Boolean function;

nondeterministic logic means for generating a first set of random boolean values of a series of variables as a first proposed solution to the stored computing problem, the nondeterministic logic means including a series of semiconductor nondeterministic logic elements, each nondeterministic logic element having a pair of outputs arranged for generating a respective one of the random boolean values and its logical complement, and the series of nondeterministic logic elements being coupled together and arranged so as to assert both the uncomplemented and complemented outputs for all the variables to a first predetermined logical state in response to receiving a common flip control signal asserted to a predetermined logical 'reset' state, and arranged so as to generate the random boolean values for all of the variables substantially concurrently in response to the flip control signal switching from the 'reset' state to a logically complementary 'generate' state;

configurable logic means having the pairs of outputs from the nondeterministic logic means, as input pairs and having a single output and including means for computing the Boolean function of the input pairs;

US 6,463,422 B1

15

the configurable logic means including a circuit which computes the Boolean function such that the output of the configurable logic means assumes a first predetermined logical value—a ‘satisfied’ value—when the flip control signal is in the logical ‘reset’ state and both outputs in every output pair of the non-deterministic logic means representing the true and complemented random variables are asserted to a first predetermined logic state;

the output of the configurable logic means assuming a logically complementary ‘unsatisfied’ value when the set of random Boolean values generated by the non-deterministic logic means in response to assertion of the flip control signal to the ‘generate’ value is not a solution to the computing problem represented by the Boolean function;

the output of the configurable logic means further assuming the ‘satisfied’ value when the set of random Boolean values generated by the non-deterministic logic means in response to assertion of the flip control signal to the ‘generate’ value is a solution to the computing problem represented by the Boolean function; and

the configurable logic means further arranged such that if the output of the configurable logic means changes from the ‘satisfied’ value to the ‘unsatisfied’ value when one output pair of the non-deterministic logic means changes to the state in which the outputs have random complementary values in response to assertion of the single flip signal to the ‘generate’ value, the configurable logic means output cannot change back to the ‘satisfied’ value as each of the remaining output pairs of the non-deterministic logic change to the state in which those outputs have a random complementary values in response to the assertion of the single flip signal to the ‘generate’ value; and

feedback means coupled to the output of the configurable logic means for asserting the flip signal to the ‘generate’ state to generate an alternative set of random boolean values as an alternative proposed solution to the computing problem responsive to the output of the configurable logic means assuming the ‘unsatisfied’ value so that the nondeterministic logic means, the configurable logic means and the feedback means together form a hardware loop operable asynchronously.

16

9. A hardware probabilistic computing system according to claim 8 wherein:

the memory means includes a plurality of rows of memory cells, each row arranged for storing a series of data bits corresponding to a respective clause of the computing problem and comprising a series of bits, each bit such corresponding to a respective one of the variables;

the configurable logic means includes, for each row of the memory means, first logic means coupled to the corresponding row of the memory means and coupled to the nondeterministic logic means for providing a respective computed function signal that indicates whether the series of random boolean values satisfies the corresponding clause of the computing problem; and

second logic means coupled to receive all of the computed function signals for determining and indicating that the first proposed solution satisfies the computing problem when all of the computed function signals indicate that the corresponding clauses of the problem are satisfied.

10. A hardware probabilistic computing system according to claim 9 wherein:

the first logic means comprises a series of crosspoint switch circuits, each crosspoint switch circuit being coupled to receive a respective bit of the corresponding row of the memory means;

each crosspoint switch circuit further being coupled to receive the corresponding variable signal from the nondeterministic logic means;

and each crosspoint switch circuit including means for coupling the corresponding probabilistic variable signal to a common circuit node only if the said respective bit has a first predetermined logic state; and

the first logic means further comprises a logic gate coupled to the common circuit node for asserting the computed function signal responsive to such of the probabilistic variable signals as are coupled to the common circuit node by the crosspoint switch circuits so as to indicate whether the first set of random Boolean values satisfies the corresponding clause of the computing problem.

* * * * *

**United States Court of Appeals
for the Federal Circuit**

15-1293

RICKY D. HANGARTNER.,

Plaintiff – Appellant,

v.

INTEL CORPORATION, INC.,

Defendant – Appellee.

CERTIFICATE OF SERVICE

Being duly sworn according to law, and being over the age of 18,
upon my oath I depose and say that:

On the date indicated below, I electronically filed the foregoing
with the Clerk of the Court for the United States Court of Appeals for
the Federal Circuit by using the appellate CM/ECF system.

I certify that all participants in the case are registered CM/ECF
users and that service will be accomplished by the appellate CM/ECF
system.

Dated: April 30, 2015

/s/ John Whitaker

CERTIFICATE OF COMPLIANCE

I hereby certify that this brief complies with the type-volume limitation of Federal Rule of Appellate Procedure 32(a)(7)(B).

This brief contains **3412** words, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(a)(7)(B)(iii) and Federal Circuit Rule of Appellate Procedure 32(b). The word count was performed by the automated word-counting function of counsel's word processing software.

This brief complies with the typeface requirements of Federal Rules of Appellate Procedure 32(a)(5,6). This brief has been prepared in a proportionally spaced typeface using LibreOffice in a 14 point "Century Schoolbook" font.

Dated: April 30, 2015

Respectfully submitted,

/s/ Philip P. Mann

Philip P. Mann